FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

DEPARTMENT OF COMPUTER SCIENCE

Thesis: Bachelor of Science in Computer Science

Information Bottleneck for Deep Learning: Theory and Experiments

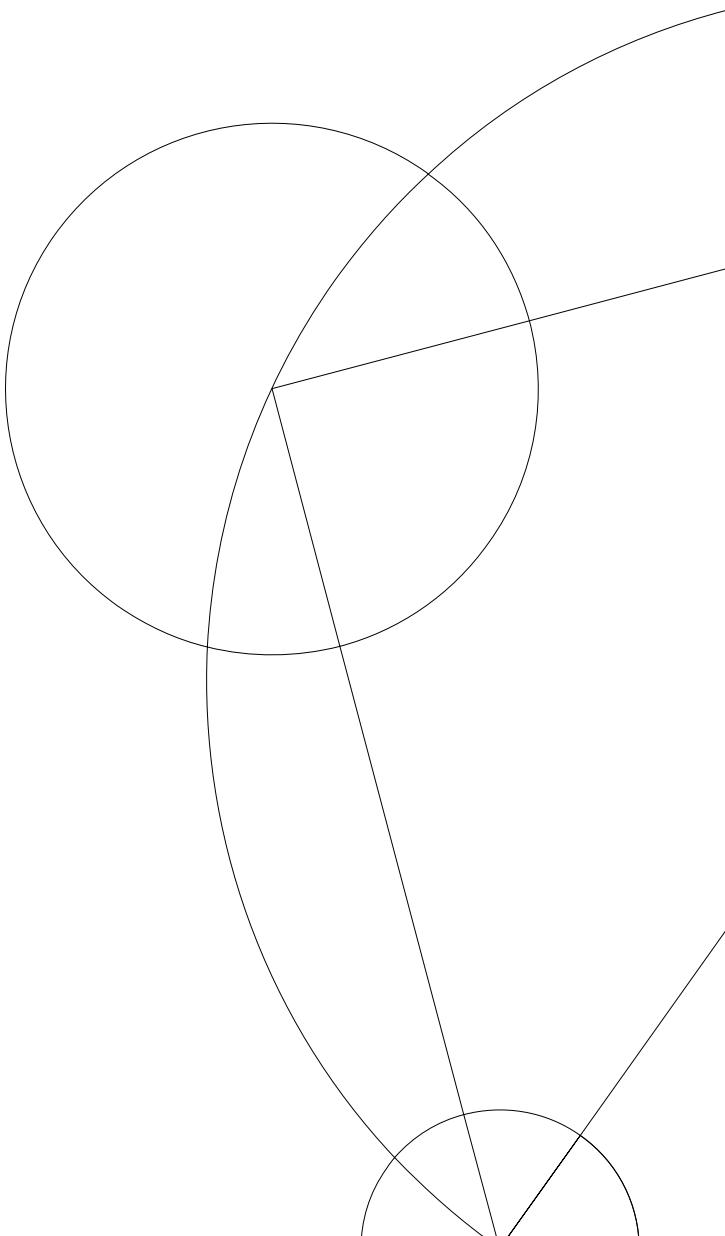
Author:

Nichlas Langhoff Rasmussen <rvq120@alumni.ku.dk>

Supervisors:

Stephan Sloth Lorenzen <lorenzen@di.ku.dk>

Christian Igel <igel@di.ku.dk>



Abstract

While Deep Neural Networks (DNN) have shown increasingly impressive results for practical applications the theoretical understanding of the successes remains largely unresolved. One recent approach to understanding DNNs is through the lens of the *Information Bottleneck* (IB) principle in an attempt to model the compression between the hidden layers and the input and desired output of a DNN. Empirical evaluations of IB for DNNs has been widely discussed and criticized, due to the inconsistencies in the results. The main objective of this thesis is to shed light on some of the central aspects of this discussion.

We present the relevant background theory related to DNNs, information theory, and the IB principle. We then investigate one of the most criticized claims: that the network training happens in two phases: the *fitting* and *compression* phase. This is done by: (i) Replicating some of the main results from the original papers that sparked this discussion. (ii) Further investigation of the claims and results by including additional activation functions and comparing methods used to estimate mutual information. We are able to reproduce the original results, but find the them unconvincing. We show that: (i) The methods used to estimate mutual information is salient to the results. (ii) Both bounded and unbounded activation functions may show compression (iii) Initialization of parameters has a large effect on the results.

Contents

1. Introduction

In later years *deep neural networks* (DNN) have shown increasingly impressive results on a variety of machine learning related tasks, pushing the boundaries in fields ranging from natural language processing [6] and image recognition [28] to reinforcement learning [21], by solving previously unsolved problems, or outperforming classic machine learning models and algorithms.

Despite these results, they are still often considered *black-box* models [1], and it is of great interest to illuminate some of the intricacies and understand *why* these models perform so well. In recent years there has been made efforts to analyze DNNs, by applying information theoretic concepts, specifically through the lens of the *Information Bottleneck* (IB) method first proposed by Tishby et al. [26]. The IB method can be thought of as a generalization of rate distortion theory (RDT) first described by Shannon [18] and further investigated in [19]. RDT addresses the fundamental trade-off in lossy compression between the *rate* and *distortion* of a signal going through a channel. In RDT the relevance of the transmitted signal is determined by a *distortion-function* however, choosing such a function is not an easy task [26]. The IB method proposes a method of generalizing this trade-off by introducing a *relevance variable*. This relevance variable should contain the information that is relevant relative to the signal being transmitted. Given this relevance variable, we now face a trade-off between compression and keeping the desired amount of relevant information in the compressed representation.

Tishby and Zaslavsky [25] first formulated the connection between the IB method and DNNs and proposed that DNNs face a similar trade-off to the IB method of compressing the input while keeping the information about the desired output. Shwartz-Ziv and Tishby [20] later confirmed this by running numerical experiments concluding, among other things, that DNNs undergo two distinct phases during training; an initial (fast) fitting phase and a (slow) compression phase. Furthermore they claim that these two phases are causally related to the gradient dynamics and the generalization of the network.

The analysis by Shwartz-Ziv and Tishby were only done on networks using the `tanh` activation function for all hidden layers, except the last. Saxe et al. [16] reproduced the original experiments and extended their analysis by experimenting with full-batch training and with the commonly used `ReLU` activation function. They concluded that no compression phase was observed in the `ReLU` networks they tested and claimed the compression phase observed in [20] is due to the double saturating activation function used, and not a general property of the training process of the network. This was followed by an extensive discussion on the methods used to estimate mutual information in the paper, casting doubt on the results [1]. Eventually Noshad et al. [14] and Chelombiev et al. [4] showed that the methods used previously may

[1] Much of the discussion is summarized here: `https://openreview.net/forum?id=ry_WPG-A-`

have underestimated the compression, and claimed that unbounded activation functions *can* indeed compress the input as well.

This thesis will start by providing the necessary theory from deep learning and information theory. In Chapter 2 we account for the basics of deep neural networks by covering the building blocks of feedforward neural networks. In Chapter 3 we provide the necessary information theoretic background including entropy, Kullback-Leibler divergence, and mutual information. In Chapter 4 we cover RDT and address some of its shortcomings before diving into the details of IB theory and its connection to DNNs. In Chapter 5 we present our experimental setup and reproduce key results from the main papers by Shwartz-Ziv and Tishby [20] and Saxe et al. [16]. After presenting the results we will discuss our findings. We highlight possible shortcomings of the mutual information estimation scheme used in the original papers and conclude that further analysis is needed. We proceed to perform additional experiments with other types of mutual information estimation methods and additional activation functions. Lastly, we discuss the results of the further experimentation before moving on to a conclusion and future work in Chapter 6.

1.1 Preliminaries and Notation

All mentions of $\log$ use base 2. Uppercase letters $(X, Y, \ldots)$ denote the names of random variables, and lowercase letters $(x, y, \ldots)$ denote the realization of the random variables. Calligraphic letters $(\mathcal{X}, \mathcal{Y}, \ldots)$ denote the sample space for the corresponding random variables i.e $x \in \mathcal{X}$. We will only consider discrete random variables i.e $\mid \mathcal{X} \mid < \infty$ where $\mid \mathcal{X} \mid$ denote the cardinality of $\mathcal{X}$. Independence between two random variables X and Y is denoted with $X \perp\!\!\!\perp Y$. We denote the probability mass function $Pr[X = x]$ as $p(x)$. $X \sim p\left(x\right)$ indicate a random variable X drawn from a distribution $p\left(x\right)$. $I\left(X;Y\right)$ denote the mutual information between two random variables X and Y. $D\left(p \mid\mid q\right)$ denote the Kullback-Leibler divergence between two distributions p and q with probability mass functions $p(x), q(x)$ respectively. We use the notation that $\{x^{(1)}, \ldots, x^{(N)}\} = \{x^{(i)}\}_{i=1}^{N}$. $\sum_x$ is used shorthand for $\sum_{x \in \mathcal{X}}$. For $X \sim p(x)$, by $E_{p(x)}[f(X)]$ we denote the expectation of $f(X)$ wrt. $p(x)$. If $p(x)$ is clear from context, we may write $E[f(X)]$. Matrices are denoted with bold uppercase letters $(\mathbf{X}, \mathbf{Y}, \ldots)$ and vectors with bold lowercase letters $(\mathbf{x}, \mathbf{y}, \ldots)$.

2. Deep Neural Networks

This chapter describes the theory for the simplest type of DNN: the feedforward neural network, which is the network considered by [16, 20], and which we will restrict our scope to in this thesis. In the rest of the thesis, we will use the terms FNN and DNN interchangeably to refer to a feedforward neural network. This is to stay consistent with the terminology used by relevant literature on the subject of Information Bottleneck theory for DNNs.

Section 2.1 describes the different components of a FNN, while Section 2.2 focuses on the so called *loss* and *cost function* of the network. Lastly, Section 2.3 describes the process of *stochastic gradient descent* in relation to FNNs. The chapter is mainly inspired by Chapters 6 and 8 in [8]

2.1 Structure of Feedforward Networks

We consider the case of supervised learning, specifically classification, where we are given a dataset containing N training samples $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$, where $\mathbf{x}^{(i)}$ are the input features of sample i and $\mathbf{y}^{(i)}$ is the desired output. $\mathbf{y}^{(i)}$ is a one-hot encoded vector i.e has a 1 in the index representing the desired output class, and 0 everywhere else. The goal is for the FNN to approximate the function $f(\mathbf{x})$ that represents our samples. This is typically done by optimizing a set of parameters $\boldsymbol{\theta}$ through *stochastic gradient descent*. $\boldsymbol{\theta}$ is a set of weight matrices and biases, $\boldsymbol{\theta} = \{\mathbf{W}^{(i)}\}_{i=1}^{L} \cup \{\mathbf{b}^{(i)}\}_{i=1}^{L}$ [8].

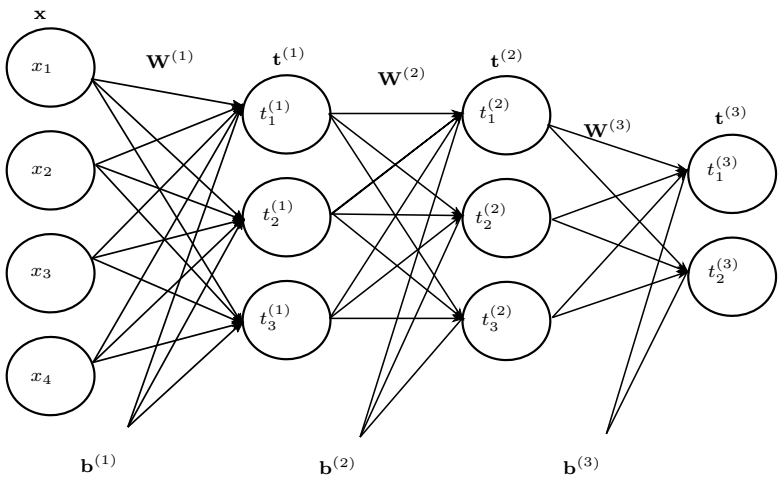


Figure 2.1: 3 Layer FNN

A FNN is made up of multiple *layers*. An illustration of a 3 layer FNN is shown in Figure 2.1. There are three types of layers in these networks. (i) The *input layer*, $\mathbf{x}$, a representation of an input sample. This layer is typically not counted when counting the layers in a network. (ii)

The *hidden layers*, $\mathbf{t}^{(l)}, l = 1, 2 \dots L$, which there can be an arbitrary number of. These hidden layers serve as intermediate representations of the input, between the input and output layer. We call them hidden layers as they will never directly see the input or expected output from the training samples. We start the hidden layer at $l = 1$, however we will use the convention $\mathbf{x} = \mathbf{t}^{(0)}$. The more hidden layers the deeper the network, and higher dimensionality of a hidden layer indicate a wider network. (iii) Likewise the *output layer* $\hat{\mathbf{y}}$ is often simply denoted as the last hidden layer namely $\mathbf{t}^{(L)}$ - as it is the case in Figure 2.1 [8].

The layers consist of nodes, that we call neurons. Each of these neurons contain a single value representing the so called *activation* of that neuron. In order to understand the computations in the *forward pass* we have to understand how these neurons act. Typically these activations are computed in two steps. Let us start by considering a single neuron in the network; The first step is the following weighted sum of the activations from the previous layer

$$z_k^{(l)} = \sum_j W_{kj}^{(l)} t_j^{(l-1)} + b_k^{(l)} \tag{2.1}$$

Here we consider $x_j = t_j^{(0)}$. (2.1) is a linear transformation of the activations from the previous layer. Even with multiple hidden layers, we will not be able to model any non-linear functions as multiple linear transformations may be viewed as a single linear transformation [8].

In order to model non-linear functions we need to add a non-linearity to (2.1). Such a non-linearity is called an *activation function*. Common choices of activation functions include the hyperbolic tangent $\tanh(x)$ and the rectified linear unit (`ReLU`) $R(x) = \max(0, x)$. For classification problems $\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$ is often used for the output layer, as it produces a valid probability distribution as the output vector, where each element is the probability of belonging to a class represented by the index. We will denote an arbitrary activation function with $\sigma(\cdot)$. By applying such a non-linearity to (2.1) we will consequently be able to model non-linear functions as well. This means that

$$t_k^{(l)} = \sigma\left(z_k^{(l)}\right)$$

Rather than considering the neurons individually we can vectorize the computations such that [8]

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)} \mathbf{t}^{(l-1)} + \mathbf{b}^{(l)}$$
$$\mathbf{t}^{(l)} = \sigma\left(\mathbf{z}^{(l)}\right)$$

2.2 Objective Functions

Typically the parameters are randomly initialized according to some distribution, eg. a Gaussian distribution, when we begin the training phase of our network. To learn the best parameters we need to specify a *loss function* and consequently a *cost function*, that we aim to minimize, such that the parameters converge to some (hopefully) optimal setting. Often the terms loss and cost function are used interchangeably [8], however we define the cost function as the average of all the loss functions. The loss function computes the loss of a single sample.

There exists many different loss function however, as we consider only classification problems we restrict our discussion to the *cross entropy* loss function. Thus we aim to find a set of parameters that minimize the cross entropy of the training data and the models learned distribution.

Let us denote the one-hot encoded label as $\mathbf{y}$ and the prediction as $\hat{\mathbf{y}}$.

$$\mathcal{L}\left(\hat{\mathbf{y}}, \mathbf{y}\right) = -\sum_{k}^{K} y_k \log(\hat{y}_k)$$

where $\mathbf{y}$ and $\hat{\mathbf{y}}$ are vectors of size K where K is the number of classes. $\hat{\mathbf{y}}$ is a vector with each index being the probability of belonging to the class that index represents. This is our loss function. Assuming we have a dataset of size N the cost function will thus be

$$C\left(\boldsymbol{\theta}\right) = -\frac{1}{N}\sum_{i}^{N} \mathcal{L}(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$$

2.3 Optimization

A *forward pass* is a pass through the network from input to prediction and then computing the cost. Now that we have defined a cost function we need to minimize this cost function wrt. $\boldsymbol{\theta}$ i.e $\min_{\boldsymbol{\theta}} C\left(\boldsymbol{\theta}\right)$. A common way of doing this is via *stochastic gradient descent* (SGD). Typically this is also called mini-batch gradient descent, as we compute an estimate of the gradient of the cost function wrt. the parameters after a small number of training examples (the mini-batch) [8]. Assuming a mini-batch of size s the cost function will thus be

$$C\left(\boldsymbol{\theta}\right) = -\frac{1}{s}\sum_{i}^{s} \mathcal{L}(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$$

At the end of the forward pass of the mini-batch we compute the gradient of the loss function wrt. the parameters in our network and update them accordingly. Specifically at update $\tau + 1$ we compute the gradients of the parameters from update τ

$$\nabla C(\boldsymbol{\theta}_\tau) = \left(\frac{\partial C}{\partial \mathbf{W}^{(1)}}, \ldots, \frac{\partial C}{\partial \mathbf{W}^{(L)}}, \frac{\partial C}{\partial \mathbf{b}^{(1)}}, \ldots, \frac{\partial C}{\partial \mathbf{b}^{(L)}}\right)$$

and update the parameters

$$\boldsymbol{\theta}_{\tau+1} = \boldsymbol{\theta}_\tau - \alpha \nabla C\left(\boldsymbol{\theta}_\tau\right)$$

Where α is the learning rate indicating how large a step we want to take in the direction of the gradients. This process produces noisy estimates of the gradient as it is approximating the gradient by only considering a mini-batch. The idea is illustrated in Figure 2.2. An alternative to this is to use *batch gradient descent* (BGD). This is equivalent to letting the mini-batch be the size of the whole training dataset. In this case we will thus only update the parameters after computing the loss for all training samples. This method produces a cleaner gradient, but is often infeasible in real world problems, as it requires us to keep the whole training dataset in memory [8].

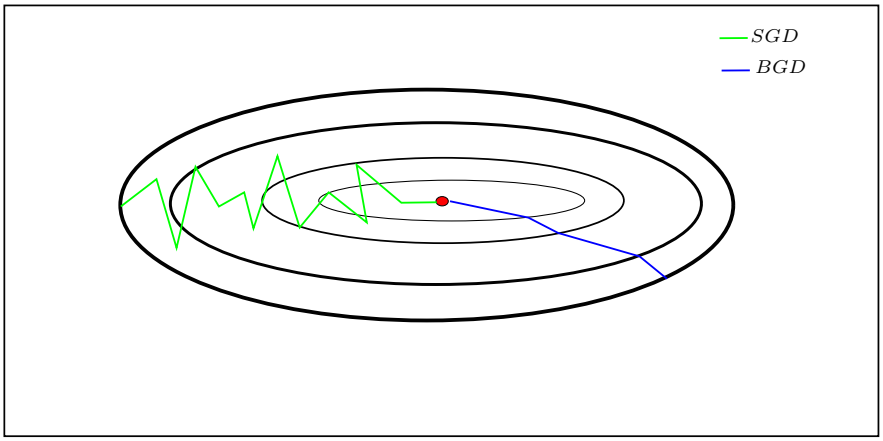


Figure 2.2: Illustrative figure of the noisy gradient process when finding a minima (indicated by the red dot) with SGD versus BGD.

3. Information Theory

This chapter introduces concepts from information theory relevant to Information Bottleneck theory. Section 3.1 introduce the basic concepts of entropy while Section 3.2 builds upon these concepts and introduces the Kullback-Leibler divergence. Lastly in Section 3.3 we introduce the concepts of mutual information along with some important related properties. The chapter is based mainly on Chapter 2 in [5].

3.1 Entropy

Given a discrete random variable $X \sim p(x)$ the *entropy* of X is defined as

$$H(X) = -\sum_x p(x) \log p(x). \tag{3.1}$$

Here we assume that $0 \log 0 = 0$ which is justified by continuity [5]. Specifically we have that $\lim_{x \to 0} xlogx = 0$ which can be evaluated by L'Hospital's Rule. Note that $H(X) \geq 0$ as $0 \leq p(x) \leq 1 \implies -\log p(x) \geq 0$.

Entropy is a measure of the average uncertainty in a random variable. It is commonly used to ask questions like, what is the average number of bits a sender need to transmit to a receiver, to describe some random variable X? How many yes/no questions are needed to determine the realization of a random variable? The more bits that needs to be transferred the more information is needed to describe the random variable, and the entropy is consequently high. Hence entropy may be seen as a measure of information in a random variable.

Consider an example where we are given a random variable, X, drawn from a Bernoulli distribution, $X \sim Bern(p)$, where $Pr[X = 1] = p, \quad Pr[X = 0] = 1 - p$, then the entropy $H(X)$ is given as

$$H(X) = -\sum_x p(x) \log p(x) = -p \log p - (1-p) \log(1-p) \coloneqq H(p) \tag{3.2}$$

It is intuitive, as seen in the Figure 3.1, that when $p \in \{0, 1\}$, there is no uncertainty about the outcome of the random variable, and the entropy is minimized. The uncertainty, and the entropy, is conversely maximized when $p = \frac{1}{2}$ [5].

Given two random variables X and Y, with a joint probability mass function $p(x, y)$. We can consider (X, Y) as a vector-valued random variable [5], and consequently the definition of joint entropy relative to (3.1) is easily introduced as,

$$H(X, Y) = -\sum_{x,y} p(x, y) \log p(x, y)$$

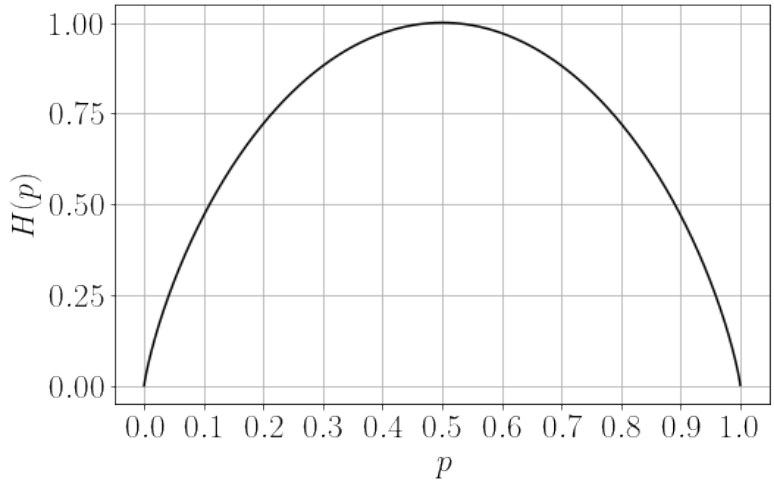


Figure 3.1: A plot of the entropy $H(p)$ as a function of p as described in (3.2). The figure is a replication of [5, Figure 2.1]

The conditional entropy of X and Y is defined as

$$\begin{aligned} H(Y|X) &= \sum_x p(x) H(Y|X=x) \\ &= -\sum_x p(x) \sum_y p(y|x) \log p(y|x) \\ &= -\sum_{x,y} p(x,y) \log p(y|x) \end{aligned}$$

Which tells us how much the expected uncertainty about Y is reduced by having observed X. A natural extension is then to introduce the chain rule for entropy, specifically the joint entropy of X and Y is given as the entropy of X plus the conditional entropy of X given Y,

$$H(X,Y) = H(X) + H(X|Y)$$

3.2 Kullback-Leibler Divergence

A quantity closely related to entropy is the *Kullback-Leibler divergence* (KL divergence). Given two distributions p and q with probability mass functions $q(x)$ and $p(x)$ the KL divergence is a measure of how inefficient it will be to assume that the underlying distribution is q rather than p. It is sometimes called the KL distance as it measures the *distance* between the distributions p and q however, it is not a true distance since it is not symmetrical and does not satisfy the triangle inequality [5, p. 19].

The KL divergence between p and q is defined as

$$D(p \mid\mid q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{3.3}$$

Thus the KL divergence is only zero when the distributions are equal, since $\log(1) = 0$. We assume that $0 \log \frac{0}{q} = 0$ and $0 \log \frac{p}{0} = \infty$ [5]. As for entropy it is easy to see that $D(p \mid\mid q) \geq 0$.

Given two joint distributions p and q with probability mass functions $p(x, y)$ and $q(x, y)$ the *conditional KL divergence* is given as the divergence between p and q for all values of $x \in \mathcal{X}$ and averaging over these values of x. Specifically,

$$D(p(y|x)) \mid\mid q(y|x)) = \sum_x p(x) \sum_y p(y|x) \log \frac{p(y|x)}{q(y|x)}.$$

3.3 Mutual Information

We will now go on to define the *mutual information* between random variables. The mutual information between random variables X and Y with a probability mass function $p(x, y)$ is defined as

$$I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} = D(p(x,y) \mid\mid p(x)p(y)) \tag{3.4}$$

The mutual information measures the information in X that is contained in Y i.e if we know Y how much do we reduce the uncertainty of X. The mutual information is the intersection in Figure 3.2. Thus if X and Y are independent then $I(X;Y) = 0$, since they share no information there is no reduction in uncertainty of X given Y. This is also easily evaluated from (3.3) as $p(x, y) = p(x)p(y) \iff X \perp\!\!\!\perp Y$. We can also describe the mutual information using our knowledge of entropy. Specifically

$$\begin{aligned} I(X;Y) &= \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\ &= \sum_{x,y} p(x,y) \log \frac{p(x|y)}{p(x)} \\ &= -\sum_{x,y} p(x,y) \log p(x) + \sum_{x,y} p(x,y) \log p(x|y) \\ &= H(X) - H(X|Y). \end{aligned}$$

Mutual information is symmetrical thus $I(X;Y) = I(Y;X) = H(Y) - H(Y|X)$. A consequent of this is also that $I(X;X) = H(X)$ [5].

The *conditional mutual information* between the random variables X and Y when conditioned a third random variable Z is defined as

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z)$$

Lastly the chain rule for mutual information states that

$$I(X_1, X_2, \ldots, X_n; Y) = \sum_{i=1}^{n} I(X_i; Y | X_{i-1}, X_{i-2}, \ldots, X_1) \tag{3.5}$$

Proof can be found in [5, p. 24]

Figure 3.2 shows the relationship between entropy and mutual information.

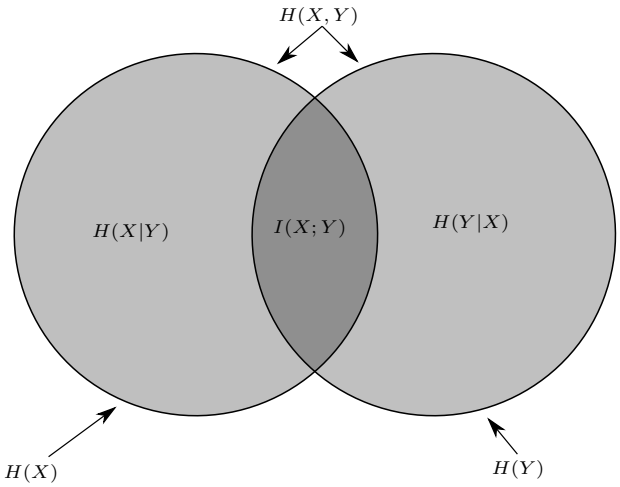


Figure 3.2: Venn diagram showing the relationships between entropy and mutual information. The figure is a replication of [5, Figure 2.2]

Given random variables X, Y, Z that form a Markov chain

$$X \to Y \to Z \iff p(x,y,z) = p(x)p(y|x)p(z|y) \tag{3.6}$$

that is X and Z are independent when conditioned on Y. The chain in (3.6) is sometimes written as $X \leftrightarrow Y \leftrightarrow Z$ as $X \to Y \to Z \implies Z \to Y \to X$ which can be shown by using the symmetry of the conditional relationship [5, p. 34]. If such a Markov chain exists then the *Data Processing Inequality* (DPI) states that

$$I(X;Y) \geq I(X;Z) \tag{3.7}$$

With equality if and only if Y is a sufficient statistic of X. This tells us that no processing of Y can increase the information Y contains about X.

Proof (see [5, p. 34-35]). Using (3.5) we have that

$$\begin{aligned} I(X;Y,Z) &= I(X;Z) + I(X;Y|Z) \\ &= I(X;Y) + I(X;Z|Y) \end{aligned}$$

From the Markov chain we have that $X \perp\!\!\!\perp Z|Y$ hence $I(X;Z|Y) = 0 \wedge I(X;Y|Z) \geq 0 \implies I(X;Y) \geq I(X;Z)$ □

Lastly we note that mutual information is invariant to invertible transformations. Specifically for any invertible and differentiable functions ϕ and ψ [2]

$$I(X,Y) = I(\phi(X), \psi(Y)).$$

4. Information Bottleneck Theory

This chapter starts off by introducing *rate distortion theory* in Section 4.1. It continues on to describe the *Information Bottleneck* principle in Section 4.2, originally presented in [26]. We will show that the Information Bottleneck principle can be thought of as a generalization of rate distortion theory, with a non-fixed distortion measure. These first two sections are mainly inspired by Chapters 7 and 10 in [5] as well as Chapter 2.1 in [22] and the original Information Bottleneck paper [26]. Lastly in Section 4.3 we connect the Information Bottleneck principle to DNNs. This section is mainly inspired by [20, 25]

4.1 Rate Distortion Theory

Let $X \sim p(x),\ x \in \mathcal{X}$ be a source random variable, and let $\hat{X}$ be a compressed representation of X, where $\hat{X}$ takes values in a reconstruction alphabet $\hat{x} \in \hat{\mathcal{X}}$. The compressed representation is thus characterized by the probability transition matrix $p\left(\hat{x} \mid x\right)$ for each $x \in \mathcal{X}$ to a *codeword* $\hat{x} \in \hat{\mathcal{X}}$ producing a soft partition of X - that is each value $x \in \mathcal{X}$ is associated with all the codewords by the normalized probability $p\left(\hat{x}\right)$ obtained by marginalizing over the joint distribution $p(x, \hat{x})$.

To determine the quality of a given compressed representation we consider two quantities from information theory. The *rate* and the *distortion*. A detailed understanding of the rate, which is typically seen as a standard measure for compression [22], is not necessary for this thesis. Instead we will focus on a strongly related quantity, $I\left(X; \hat{X}\right)$, that roughly measures how compressed our compressed representation really is [22]. Below we give some intuition on *why* this is the case. $I\left(X; \hat{X}\right)$ is maximized whenever $\hat{X}$ is a sufficient statistic of X as no information is lost. This is e.g. the case whenever $\hat{X} = X$ however, in this case no compression has taken place either. Consider conversely if we let the compressed representation take a single value, zero bit needed for the representation, then $I\left(X; \hat{X}\right) = 0$ since both $H(\hat{X}) = H(\hat{X} \mid X) = 0$ however we might have discarded all relevant information of X by compressing it this much. The general idea is illustrated in Figure 4.1. Another way is to use the arguments of the *asymptotic equipartition property* (AEP) [5, Chapter 3] to show that the maximum number of sequences we can transmit from X to $\hat{X}$ without confusion is upper bounded by $I\left(X; \hat{X}\right)$. AEP tells us that for *each* typical sequence from $\hat{X}$ of length n there are approximately $2^{nH\left(X|\hat{X}\right)}$ equally likely possible typical sequences of length n from X. However the total number of typical sequences of length n from X is approximately $2^{nH(X)}$ where each sequence has probability $2^{-nH(X)}$. In order to transmit a sequence from X to $\hat{X}$ without confusion we need to produce a maximum of $2^{n(H(X)-H\left(X|\hat{X}\right)} = 2^{nI\left(X;\hat{X}\right)}$ disjoint subsets so each partition of X map to the same codeword [22].

Thus the term $I\left(X; \hat{X}\right)$ roughly quantify how compressed our compressed representation is. However, the term is clearly not enough to determine the quality as we have no way of knowing

if we compress all the relevant information out of the random variable (consider the example where $I(X;\hat{X}) = 0$).

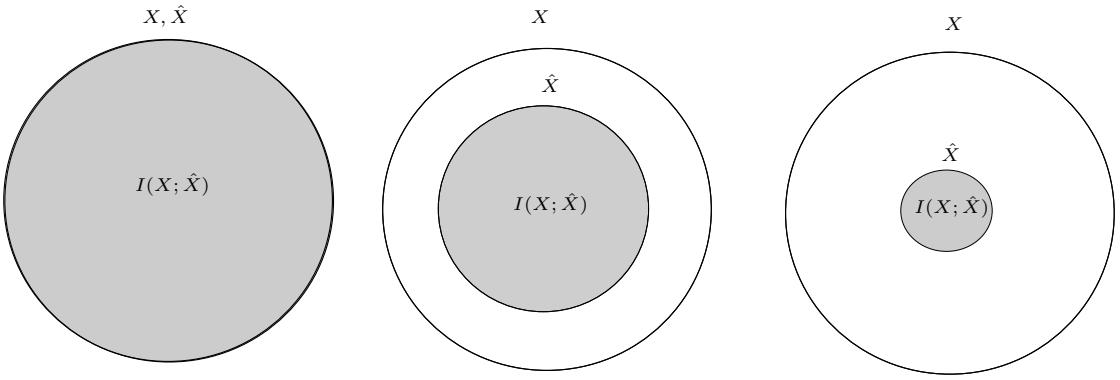


Figure 4.1: The idea of how compression and mutual information is related. The first circle show when $X = \hat{X}$ and the mutual information is maximized. The second and third circle, from the left show $\hat{X}$ as a compressed representation of X and that the mutual information $I\left(X;\hat{X}\right)$ consequently gets smaller

We address this by adding a *distortion-function* or *distortion-measure* $d : \mathcal{X} \times \hat{\mathcal{X}} \mapsto [0,\infty)$ to our compression scheme. The distortion-function is the cost of representing a symbol x by the reconstruction $\hat{x}$, and so consequently it is assumed that a smaller distortion implies a better compressed representation. Common choices of distortion-functions include the squared error distortion $d(x,\hat{x}) = (x-\hat{x})^2$ and the Hamming distortion $d(x,\hat{x}) = 1[x \neq \hat{x}]$ [5, pp. 304-305]. Given such a distortion-function, and assuming that x occurs with probability $p(x)$, and consequently the reconstruction $\hat{x}$ occurs with probability $p(\hat{x} \mid x)$ then the expected distortion is [26]

$$\mathbb{E}_{p(\hat{x},x)}\left[d\left(X,\hat{X}\right)\right] = \sum_{x,\hat{x}} p(\hat{x},x)\, d(x,\hat{x}) = \sum_{x,\hat{x}} p(x) p(\hat{x} \mid x)\, d(x,\hat{x})$$

Thus we conclude that for the quality of a compressed representation there is this trade-off between how compressed our compressed representation is and the expected distortion. This trade-off is addressed by the *rate distortion function* $R(D)$. We say that a rate distortion pair (R, D), where R is the rate and D is some distortion, is achievable if $\mathbb{E}_{p(\hat{x},x)}[d\left(X,\hat{X}\right)] \leq D$. [5]

Given a distortion function $d(x,\hat{x})$ and an i.i.d source $X \sim p(x)$ the rate distortion function is defined as

$$R(D) \equiv \min_{\{p(\hat{x}|x):\mathbb{E}[d(X,\hat{X})]\leq D\}} I(X;\hat{X}) \tag{4.1}$$

i.e we want to find the most compressed representation at a distortion level D. $p(x)$ is fixed, so we minimize over all conditionals $p(\hat{x} \mid x)$ for which the joint distribution $p(x,\hat{x})$ satisfies the distortion constraint [5, p.307].

$R(D)$ is a non-increasing convex function, and so the problem of finding the optimal map is a problem of minimizing a convex function $R(D)$ over the convex set of all possible normalized conditional distributions $p(\hat{x} \mid x)$. This can be done by introducing a Lagrange multiplier β and minimizing the functional [5]

$$\mathcal{F}[p(\hat{x}|x)] = I(X;\hat{X}) + \beta\mathbb{E}[d(X,\hat{X})]_{p(x,\bar{x})}$$

The solution to the functional equation

$$\frac{\partial \mathcal{F}}{\partial p\left(\hat{x} \mid x\right)}=0.$$

where $\sum_{\hat{x}} p\left(\hat{x}|x\right)=1$ is given by

$$p(\hat{x}|x)=\frac{p(\hat{x})}{Z(x,\beta)}\exp[-\beta d(x,\hat{x})] \tag{4.2}$$

Where $Z\left(x,\beta\right)$ is a normalization factor [5]. Note that $p\left(\hat{x}\right)$ is obtained by marginalizing over the joint distribution $p\left(x,\hat{x}\right)$ and so it depends on $p\left(\hat{x} \mid x\right)$. Actually computing the rate distortion function requires an iterative algorithm like the Blahut-Arimoto algorithm; this is not strictly crucial for our understanding however, a detailed discussion may be found in [26].

The problem of choosing a "good" distortion-function still remains. The distortion-function essentially aims to capture the relevant information of the input variable such that the input can be compressed as much as possible while still keeping the relevant information. Clearly different distortion measures may lead to different rate distortion functions, so without choosing a relevant measure it can be difficult to determine the quality of our result. The Information Bottleneck principle attempts to solve this issue.

4.2 The Information Bottleneck Principle

The Information Bottleneck principle [26] presents a way of dealing with this problem of choosing a distortion-function, by introducing a *relevance variable* Y. The relevance variable Y should define the information we find relevant w.r.t. X. Consider a prediction problem of determining whether a picture is of a cat or a dog. We are given some input X, which is a representation of the picture. The target variable defines what information we find relevant based on the input. In our case, the target variable $Y \in \{\text{cat}, \text{dog}\}$. As proven by Shamir et al. [17] the mutual information $I\left(\hat{X};Y\right)$ is causally related to generalization error for learning problems. Hence it is desirable to obtain a compressed representation of X, $\hat{X}$ that keeps only the relevant features w.r.t. to Y, and discards of the irrelevant information. For classification problems this will hopefully yield an easily separable manifold relative to the classes. Note that, as it is clear from the example, the input variable X and the relevance variable Y must have positive mutual information (be statistically dependent). Instead of only assuming access to the distribution $p(x)$ as it is the case in rate distortion theory we assume access to the joint distribution $p\left(x,y\right)$ [1].

More formally the idea of the Information Bottleneck principle is to compress the input X into some representation $\hat{X}$. $\hat{X}$ should contain as much information as possible w.r.t the relevance variable Y while discarding as much irrelevant information as possible. The compressed representation $\hat{X}$ should only be dependent on X and not Y. Consequently we have a Markov chain $Y \rightarrow X \rightarrow \hat{X}$. Hence the compressed representation must satisfy

$$I(\hat{X};Y) \leq I(X;Y)$$

Similar to the trade-off in rate distortion theory, we have a trade-off of compressing the input X and maintaining the relevant features in X w.r.t Y. Thus as in rate distortion theory we would

[1] This is a large point of criticism for the IB method relative to learning problems such as the ones we will face with DNNs. We are aware of this, but do not dive more into it in this thesis. The reader is referred to [17], where it is shown that the information bottleneck can find good representations with good generalizations even when the underlying distribution is estimated from finite samples much smaller than the true distribution.

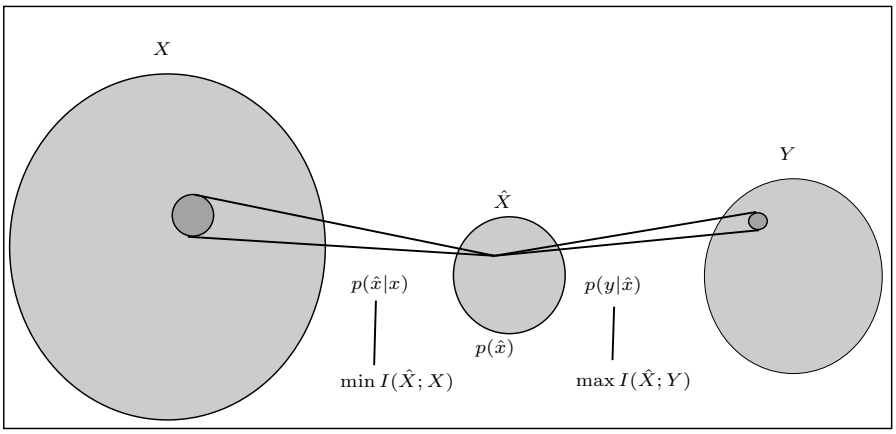


Figure 4.2: The IB setup visualized. The joint distribution $p\left(x,y\right)$ is given and the rest are a visualisation of the self-consistent equations we need to iterate to obtain a solution, as seen in (4.5). We want to find a encoder $p\left(\hat{x}|x\right)$ that minimize the mutual information $I\left(\hat{X};X\right)$ and a decoder $p(y|\hat{x})$ that maximize $I\left(\hat{X};Y\right)$. The figure is a replication of [22, Figure 2.5]

like to minimize the rate $I(\hat{X};X)$, but rather than minimizing the expected distortion we now want to maximize the relevant information $I(\hat{X};Y)$. The idea is visualized in Figure 4.2. We can express this trade-off through a *compression-relevance function* similar to the rate distortion function in rate distortion theory [22]

$$\hat{R}\left(\hat{D}\right) = \min_{\{p(\hat{x}|x):I\left(\hat{X};Y\right)\geq\hat{D}\}} I\left(\hat{X};X\right) \tag{4.3}$$

That is, we want to minimize the rate of the compression $I(\hat{X};X)$ under the constraint that the relevant information is greater than some threshold $\hat{D}$, $I(\hat{X};Y)\geq\hat{D}$ subject to the Markov constraint. The solution to this problem can be expressed as a minimization of the functional [25]

$$\mathcal{L}[p(\hat{x}|x)] = I(\hat{X};X) - \beta I(\hat{X};Y) \tag{4.4}$$

Where the Lagrange multiplier β is the trade-off parameter of minimizing $I(\hat{X};X)$ and maximizing $I(\hat{X};Y)$. If we take the extreme cases where $\beta\to 0$ we map everything to a single cluster and thus maximize the compression, however we have also compressed the input so much that possibly no information is kept w.r.t. Y. On the other hand if we let $\beta\to\infty$ we maximize the mutual information between $\hat{X}$ and Y but no compression has taken place and $\hat{X}=X$ [22]. The interesting values of β are somewhere in-between these two extremes. The trade-off is depicted by the black curve, called the IB curve in Figure 4.3.

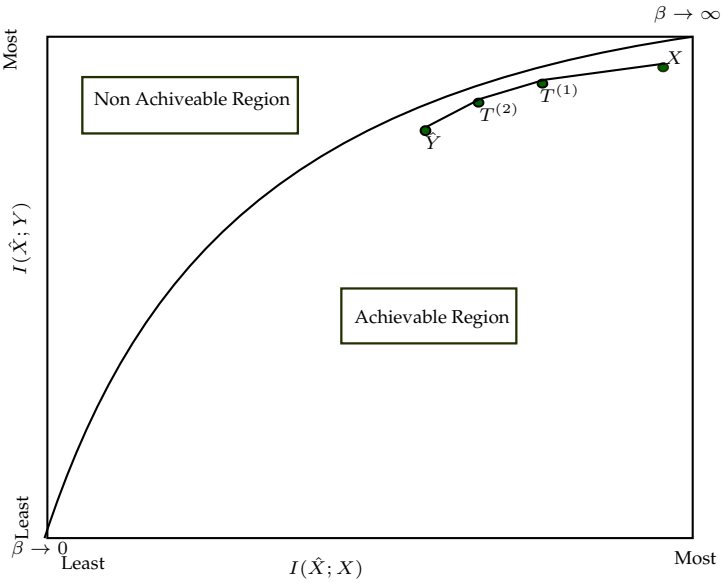


Figure 4.3: An illustration of the IB curve/IB limit depicted by the black curve in the figure. The black curve is a monotonic decreasing positive curve and has a slope of β^{-1}, for smooth joint distributions $p(x, y)$ [25]. The line connected by the dots show a hypothesised example path for each layer in a DNN in the information plane. See Section 4.3 for further explanation on DNNs in the information plane. Figure based on [25, Figure 2]

[26] shows that the optimal solutions for (4.4) satisfy three self-consistent equations for some $\beta > 0$, namely

$$\begin{cases} p(\hat{x}|x) & = \frac{p(\hat{x})}{Z(x;\beta)} \exp(-\beta D[p(y|x)\|p(y|\hat{x})]) \\ p(y|\hat{x}) & = \sum_x p(y|x)p(x|\hat{x}) \\ p(\hat{x}) & = \sum_x p(x)p(\hat{x}|x) \end{cases} . \tag{4.5}$$

where $Z(x;\beta) = \sum_{\hat{x}} p(\hat{x}) \exp(-\beta D[p(y|x) \,||\, p(y|\hat{x})])$ is the normalization factor. Finding the optimal $p(\hat{x} \mid x)$ is a question of iterating these self-consistent equations e.g. using generalized Blahut-Arimoto like methods as presented in [22, 26]

[25] notes that the Information Bottleneck method can be interpreted as a rate distortion problem with distortion-measure $d_{IB} = D[p(y \mid x) \,||\, p(y|\hat{x})]$. The distortion-measure appears from the solution to (4.4) and is not chosen explicitly, thus it is considered the "correct" distortion-measure for this problem. For this reason the Information Bottleneck can be seen as a generalization of rate distortion theory. For the distortion-measure d_{IB} the expected distortion is [25]

$$\begin{aligned} D_{IB} &= \mathbb{E}\left[d_{IB}\left(X, \hat{X}\right)\right] \\ &= I\left(X; Y|\hat{X}\right) \\ &= I\left(X; Y, \hat{X}\right) - I(X; \hat{X}) \\ &= I(X;Y) - I(\hat{X};Y) \end{aligned}$$

Where the last equality comes from the Markov chain constraint. We can use this to rewrite (4.1) such that the functional becomes [25]

$$\hat{\mathcal{L}}[p(\hat{x}|x)] = I\left(X; \hat{X}\right) + \beta I\left(X; Y|\hat{X}\right) \tag{4.6}$$

Here we are interested in minimizing both the rate $I(X;\hat{X})$ and the expected distortion $I(X;Y|\hat{X})$. This is however, not a standard rate distortion problem as the distortion-measure is non-fixed and depends on the mapping (the decoder from the compressed space to the relevance space) $p\left(y|\hat{x}\right)$ [22].

4.3 DNNs from an IB perspective

In order to relate the IB principle to DNNs we consider each layer as a (possibly) high dimensional random variable. For an L-layer FNN denote the input layer X, the l'th hidden layer as $T^{(l)}$ and the output layer as $\hat{Y} = T^{(L)}$. The desired output of the network is a random variable Y and an arbitrary hidden layer will be denoted T.

One of the primary assumptions of using IB for DNNs is that typically the input X is some high dimensional random variable e.g. containing the pixels of an image. The label Y on the other hand is very low dimensional e.g. for binary classification it is often a one-hot encoded vector of dimension two, as described in Chapter 2. The idea is thus that much of the entropy in X carry little information relevant to the prediction of Y. Therefore if we are able to compress the input X into some intermediate representation T, we will hopefully end up with some representation of the input data that allows us to separate the data easily to predict the output [25]. It is furthermore shown by Shamir et al. [17] that the mutual information between the compressed representation T and the relevance variable Y, $I\left(Y;T\right)$, can bound the generalization error in a supervised learning setting, as it is the case for the FNNs we consider.

The connection between IB and DNN was first presented by Tishby and Zaslavsky [25]. In this paper the authors consider a classification DNN trained with stochastic gradient descent, similar to what we explained in Chapter 2. The authors note that a DNN forms a Markov chain of successive refined representations of the input layer. This can be seen as layer $l+1$ is only dependent on layer l. Specifically,

$$Y \to X \to T^{(1)} \to T^{(2)} \ldots \to T^{(L-1)} \to \hat{Y}. \tag{4.7}$$

Relative to the Markov constraint from IB theory we see that such a Markov chain exist for all hidden layers in the network as $Y \to X \to T$ [20]. Each layer in the network, may thus be characterized by an encoder $p\left(t|x\right)$ and a decoder $p\left(y|t\right)$. Consequently the representation of a given layer can be quantified by a point $(I(X;T), I(T;Y))$ in the *information-plane* [20], see Figure 5.1. This of course, once again, requires access to the joint distribution $p\left(x,y\right)$ which rarely is available for real world tasks. As mentioned previously we refer the reader to discussion on IB limits for finite samples in [17]. As a consequence of (4.7) the mutual information for the different layers in the network should follow the DPI constraint

$$I\left(Y;X\right) \geq I\left(Y;T^{(1)}\right) \geq I\left(Y;T^{(2)}\right) \ldots \geq I\left(Y;\hat{Y}\right) \tag{4.8}$$

$$I\left(X;X\right) \geq I\left(X;T^{(1)}\right) \geq I\left(X;T^{(2)}\right) \ldots \geq I\left(X;\hat{Y}\right). \tag{4.9}$$

Because mutual information is invariant to invertible transformations many different DNNs lie on the same point in the information plane (even for different architectures). Therefore we do not care about the individual neurons, but the flow of information though the layers as a whole [20]. By applying (4.4) to a DNN we can consider the IB optimal representation for each hidden layer for some value β [20]

$$\mathcal{L}\left[p\left(t|x\right)\right] = I\left(X;T\right) - \beta I\left(T;Y\right) \tag{4.10}$$

An example path for a 3 layer DNN is depicted by the dotted line in Figure 4.3. The authors suggest that layers in a DNN do in fact move towards the IB limit (see Figure 4.3) when optimized, and thus the IB trade-off seems like a natural optimization goal for DNNs [25].

5. Investigating empirical observed properties of IB in DNNs

In the following chapter we investigate and discuss some of the empirical claims regarding IB theory for DNNs. Specifically we focus on the observation and claim by Shwartz-Ziv and Tishby [20] that DNNs undergo two distinct phases visualised in the information plane during training. The first phase is the so called the *fitting phase* defined by an increase of both $I(T;Y)$ and $I(X;T)$. The second phase is the *compression phase* defined by a decrease in the mutual information $I(X;T)$. This claim has been widely discussed, among others by Saxe et al. [16] and Chelombiev et al. [4]. Saxe et al. claim that the compression phase observed in [20] is a result of using double saturating activation functions. They, among other things, claim that for unbounded activation functions such as `ReLU` no compression is observed. Chelombiev et al. [4] further investigate these claims by applying different method of estimating mutual information. They claim that compression may occur in networks trained with unbounded activation functions such as the Exponential Linear Unit (`ELU`) [1].

In Section 5.1 we start by outlining the claims from [16, 20] in detail. Next in Section 5.2 we present the general experimental setup used to carry out the numerical experiments shown in the imminent sections. In Section 5.3 we reproduce and discuss some of the key results from the papers [16, 20] related to the information plane and the observed phases. We furthermore introduce the *uniform binning* as a discretization method we use to estimate the distribution of hidden layers. In Section 5.4 we run and discuss further experiments. Firstly we investigate the effect of the maximum activation values in the information plane when estimating the mutual information using the uniform discretization approach. Next we compare the uniform discretization strategy, to an *adaptive binning* strategy used in [4]. Here adaptive refers to the intervals being depended on the distribution of the weights, rather than uniform. In the comparison we include other activation functions, namely the double saturating `ReLU6`[2] function, and the non-saturating `ELU` function.

5.1 Observed properties of DNNs in the information plane

Shwartz-Ziv and Tishby [20] empirically verify some of the ideas originally outlined in [25] and discussed in Section 4.3. They argue that the information plane is an important aspect of gaining further insight into the inner workings of DNNs. Specifically they show that the classification DNNs considered in their paper, when trained with SGD, undergo two distinct phases

[1] `ELU` is given by $g(x) = \begin{cases} x, & x \geq 0 \\ \alpha(\exp(x) - 1), & x < 0 \end{cases}$, specifically we let $\alpha = 1$.

[2] `ReLU6` is given by $r(x) = \min(\max(0, x), 6)$.

during training. These phases can be seen in Figure 5.1 depicting an illustrative information plane, as observed by the authors for a 5 layer DNN. The first phase from $A - C$ is the *fitting phase* (called the error minimization phase by Shwartz-Ziv and Tishby [20]). During the fitting phase the network is fitted to the training samples. This phase is characterised by an increase in both the mutual information of the hidden layers and the input $I(X;T)$ and the mutual information of the hidden layers and the output $I(T;Y)$. During this phase the training error rapily decrease for both the train and test data. Following this phase, the authors observe a slow phase, from $C - E$, dubbed the compression phase. During this phase the layers compress the input by decreasing $I(X;T)$, and get rid of the (hopefully) irrelevant information, as it is keeping the information about the output high and consequently keeping the generalization error low. Throughout the training process the DPI constraint remains valid. Note that no form of regularization is performed during this analysis - thus this compression phenomena is observed despite there being no specific objective for this. Rather the authors suggest that the compression occurs due to the noise induced in the optimization process by SGD [20].

Shwartz-Ziv and Tishby [20] claim that the final position of the hidden layers concentrate around the IB limit. Consequently the decoder and encoder distributions $p(t|x)$, $p(y|t)$ satisfy the IB self-consistent equations (4.5), concluding that the objective for a DNN is similar to that of the IB self-consistent equations - to find the best possible trade-off between compression and relevant information [20].

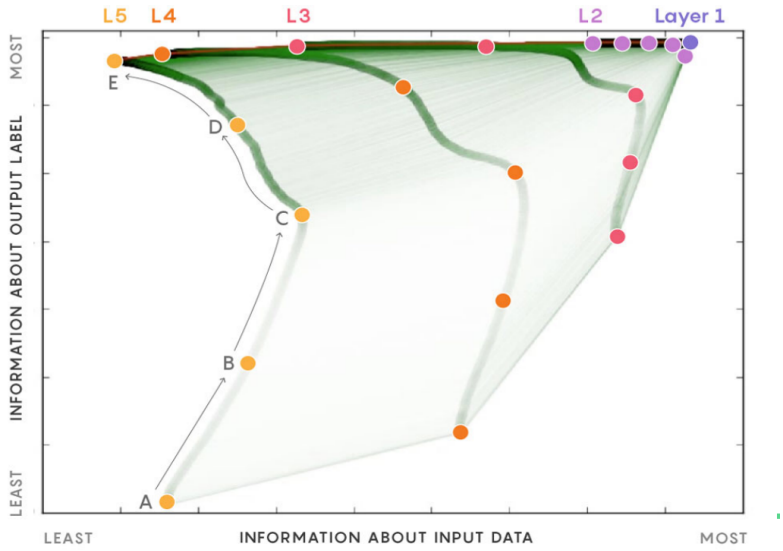


Figure 5.1: Figure of the information plane taken from [27] during training of a 5 layer network. The saturation in the green color is proportional to the number of epochs. 2 distinct phases are observed. First from $A - C$ where the layers gain information about both the output and input - this is a faster phase where the training error is minimized. From $C - E$ a slow phase is observed where the error remain low, but the network compress the representation of the input, discarding the irrelevant information about the output.

These claims are further examined by Saxe et al. [16]. The authors claim that the results from [20] do not hold in the general case, and thus cast doubt on the IB theory as a possible explanation for describing the generalization and success of DNNs. They claim the compression phase is an effect of the double saturating activation function used in [20], rather than a general property of the optimization goal.

They show that, by using a double saturating non-linearity such as the $\tanh$ function, a compression phase occurs, however by changing it to the commonly used unbounded `ReLU` function the compression phase is not visible in the information plane [16]. The authors run simulations that show these results on the original dataset used in [20] as well as the MNIST dataset.

Furthermore they claim that the compression phase is not causally related to the use of SGD. They show that training the considered networks with batch gradient descent results in similar information plane dynamics, as the networks trained with SGD. Lastly the observations by Saxe et al. [16] is consistent across all the different strategies they use to measure mutual information. This includes the uniform binning scheme, non-parametric kernel density estimators and non-parametric k-NN estimators [16].

5.2 Experimental Setup

We use the artificial dataset from the original paper by Shwartz-Ziv and Tishby [20]. The dataset consists of 4096 samples. Each sample consists of an input, a 12-bit binary vector, and an output, a 1-bit label denoting the binary class of the input. All input bit patterns in the dataset are equally likely, and labels are equally balanced (see [20] for details). We use a 80-20 split of the data where 80% is used for training, and the remaining 20% is used for testing. For consistency with previous work on the subject the information planes are based on the entire dataset [4, 16, 20]. That is, after training the network for one epoch the whole dataset is fed through the network, and these saved activation values are the values used to compute the mutual information shown in the information plane.

The network considered is a feedforward neural network, as the ones described in Chapter 2, with layers of size $12 - 10 - 7 - 5 - 4 - 3 - 2$. This is the consistent with one of networks used in [20] and the main network considered in [4, 16]. We will refer to this network structure as the `IB network`. We used Python 3.8.2 and PyTorch 1.4.0 [15] for the implementation. Code can be found in Appendix C

Adam [10] is used for training using a learning rate $\alpha = 10^{-4}$ and optimizing cross entropy. For each experiment in Section 5.3 we train two networks with the same initial weights: one network trained with mini-batches of size 256, and one network trained with a full-batch. In Section 5.4 we only use mini-batches of size 256. The activation function for the output layer will always be the `softmax` function. The weights for a hidden layer T of size $\mid T \mid$ is initialized according to a truncated Gaussian distribution with mean $\mu = 0$ and standard deviation $\sigma = \frac{1}{\sqrt{|T|}}$. The biases are initialized to zero.

Unless stated otherwise the information plane show the result of training the network for 8000 epochs for 40 runs and then taking the average over all runs. Each run is initialized randomly both with respect to the data split and the weight initialization.[3] To stay consistent with Figure 4.3 the left-most path in the information plane plot is the path of the output layer and the right-most is the path of the first hidden layer.

[3]Note: the original papers [16, 20] average over 50 run. We scaled this down due to computational limitations. According to experiments in [4] even averaging 10 runs show a clear picture of the dynamics.

5.3 Replication of Experiments

We start by investigating the two primary claims from [16, 20] related to the dynamics in the information plane. (i) The dynamics are dependent on the activation function used, and the fitting and compression phases are observed for $\tanh$ when training with a mini-batch size of 256, but *also* when trained on a full-batch size. (ii) When changing the activation function to `ReLU` only a fitting phase is observed.

After training the network we discretize the activation values for each neuron in each layer by binning the activation values into uniform bins, i.e by partitioning the output domain into M equally sized intervals. Now we treat the whole discretized layer T as a discrete random variable and estimate the mutual information for each hidden layer and the input and desired output as

$$I(X;T) = -\sum_t p(t)\log p(t) - \sum_x p(x)H(T|X=x) = H(T) - H(T|X) = H(T) \tag{5.1}$$

$$I(T;Y) = -\sum_t p(t)\log p(t) - \sum_y p(y)H(T|Y=y) = H(T) - H(T|Y) \tag{5.2}$$

The last equality in (5.1) comes from the fact that T is deterministic given X so $p\left(t|x\right) = 1 \implies H(T|X) = 0$ [4, 16]. The training process and process of estimating mutual information does not interfere with each other.

We thus consider the two settings mentioned below:

- **Setting 1.1:** In the first setting we use the $\tanh$ activation function for all hidden layers, except the last. The binning range is from $[-1, 1]$ and the number of bins are 30, as presented in [16, 20].
- **Setting 1.2:** The second setting use the `ReLU` activation function for all hidden layers, except the last. The binning range in this setting is $[0, \max(\text{activation values})]$ where $\max\left(\text{activation values}\right)$ is the maximum activation value observed. Since the mutual information is computed for each network and then averaged, this is the maximum value for a single run. We use 100 bins in this setting, as done in [16]. (See Figure 5.3)

5.3.1 Results

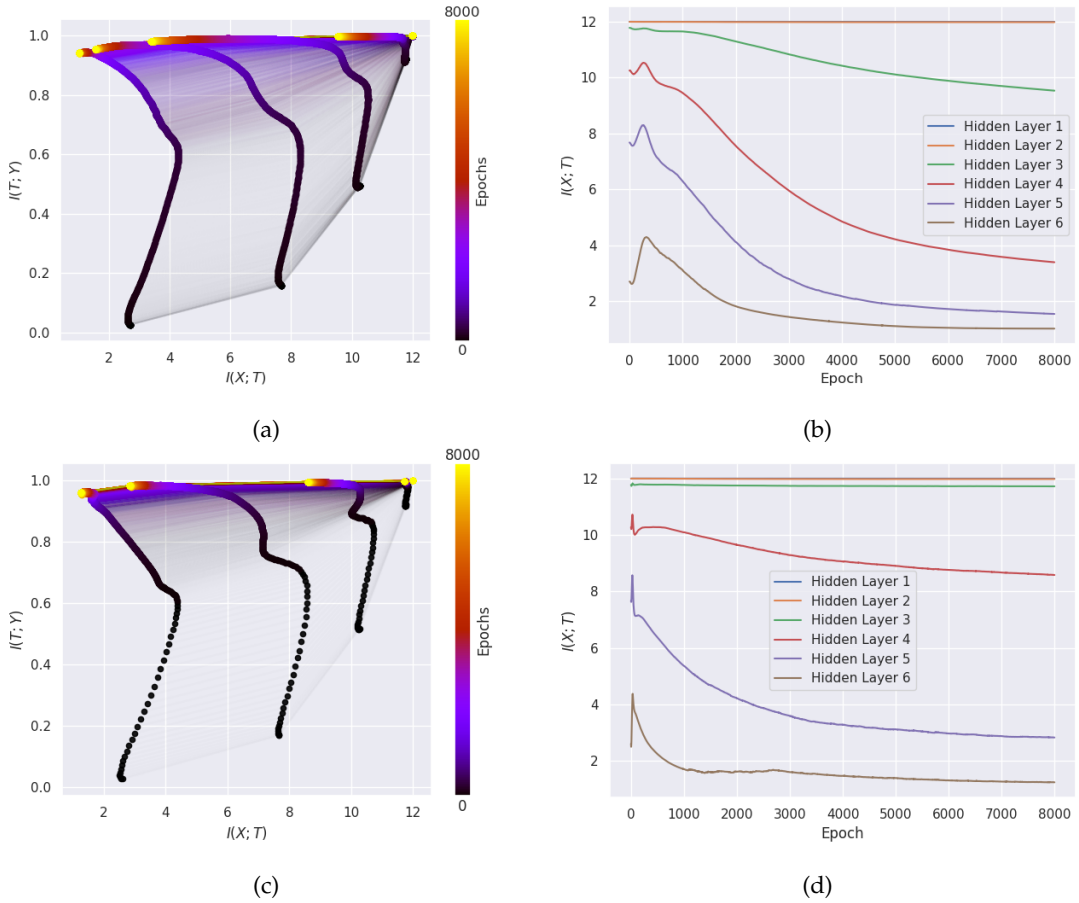


Figure 5.2: **Setting 1.1:** Visualization of the information plane and $I(X;T)$ for the `IB network` using the `tanh` activation function for all hidden layers except the last where softmax is used. (a,b) show the network when trained on a full-batch, while (c,d) show the network when trained with a batch size of 256.

Figure 5.2 shows the result of **Setting 1.1**. Both a fitting and compression phase is visible regardless of the batch size used. The phases are more apparent for most of the layers when using a full-batch size. We observe more compression, for most layers in the network trained with a full-batch, than when trained on a mini-batch.

Figure 5.3 shows the results of **Settings 1.2**. We see that only the fitting phase is observed for all the `ReLU` layers, independent of whether we used a full-batch or a mini-batch. Interestingly we see that the DPI breaks, as some of the last hidden layers contain more information about the output than the layers closer to the input. Lastly we note that the starting position of the layers in the information plane are drastically different for Setting 1.2 relative to Setting 1.1.

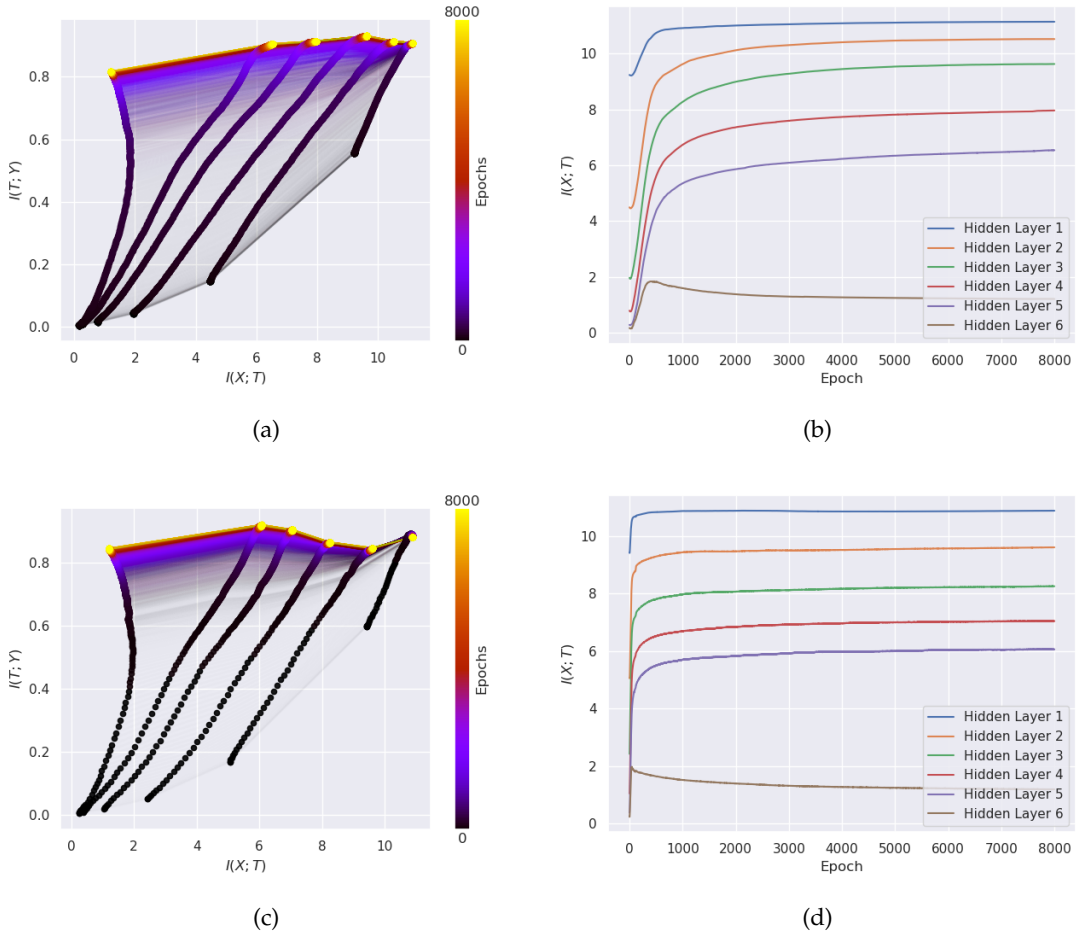


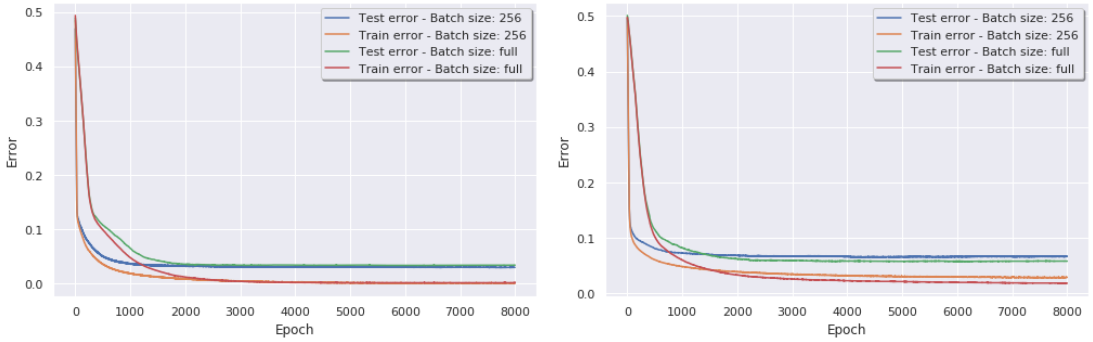
Figure 5.3: **Setting 1.2:** Visualization of the information plane and $I(X;T)$ for the `IB network` using the `ReLU` activation function for all hidden layers except the last where `softmax` is used. (a,b) show the network when trained on a full-batch, while (c,d) show the network when trained with a batch size of 256.

5.3.2 Discussion of results

As pointed our previously, both a compression phase, and a fitting phase is observed in Figure 5.2 (a,c). This is consistent with the observations made by Saxe et al. [16], and seems like an immediate contradiction to the claim by Shwartz-Ziv and Tishby [20] that the compression phase occurs due to the noisy gradients induced by the SGD when training with a mini-batch. One reason for this observation might be that the compression is not causally related to the noisy gradients, but rather an effect of using a double saturating activation function such as `tanh` [16]. However even when using a full-batch (of training data) *some* noise is still part of the training process, albeit the noise in the gradient might not be as strong as when it is estimated from the mini-batches. The fact that the full-batch case shows *more* compression, might simply be due to the fact that other parameters of the model were kept constant. Generally it has been shown that parameters such as the learning rate should change with the batch size to obtain comparable gradient dynamics [9].

To further strengthen the hypothesis that the compression phase occurs due to using a double saturating activation function, we observe that Setting 1.2 with the non (double) saturating activation function `ReLU` (Figure 5.3) does not show compression for either the mini-batch or

full-batch case. However, as we see in Figure 5.4 while it generalize slightly worse than the network from Setting 1.1, the network from Setting 1.2 still performs well. It seems excessive that the compression should only occur in this small difference between Figure 5.4 (a) and (b). This suggest that the compression is, in the general case, unrelated to the generalization of the network which is one of the original claims by Shwartz-Ziv and Tishby [20]. The problem we consider here is however, very small, and it is possible that whether we compress the representation or not will have little effect on the generalization of the network. It would be interesting to investigate whether this also holds for much larger networks.



(a) (b)

Figure 5.4: (a) shows the train and test error for **Setting 1.1**. (b) shows the train and test error for **Setting 1.2**. The error here is the 0-1 loss i.e $1 - \texttt{accuracy}$. As seen both the error on the train and test data is low regardless of the batch size for both settings. The error minimization phase is faster (in terms of epochs) for the mini-batch than the full-batch. This is consistent with the information plane shown in Figure 5.3 and Figure 5.2

While the discretization process is a necessary step to visualize the information plane, it also adds noise to the layers. This noise is not part of the actual network. Thus the actual visualization of the information plane is not exactly a correct representation of the hidden layers but nevertheless a necessity. It is unclear how to determine whether model robustness to noise like this is linked with generalization in DNNs [16].

For now we have been able to reproduce something very similar to the results from [16, 20], and generally our findings seem to side with the observations made by Saxe et al. [16] however, while experimenting we noted several things related to the results that cast doubt on the estimation of mutual information for `ReLU` layers with a uniform binning strategy.

- The initial position of the hidden layers in the information plane differ significantly between the two settings. Specifically the `ReLU` layers seem to cluster around the left corner of the information plane initially. While it makes sense that the entropy initially drops for each layer given the structure of our network (the layer widths get smaller and smaller), the clustering in one corner seems suspicious. A possible explanation is that given that the binning range is chosen such that the upper bound is $\max(\text{activation values})$ during the first few epochs when most of the activation values are very small the activation values will concentrate in few bins towards the lower end. This consequently means that the probability distribution obtained by discretizing the activation values will yield high probabilities for few possible realizations of the random variable T. This leads to low entropy and thus low mutual information, as $I(T;X) = H(T)$ and $I(T;Y) =$

$H(T) - H(T|Y)$, which leads to the results shown in Figure 5.3. This suggests that the uniform binning strategy for unbounded functions is insufficient.

- The DPI is violated in Setting 1.2 e.g $I(T^{(4)};Y) > I\left(T^{(3)};Y\right)$. A break in the theory like this, suggest that the method for mutual information estimation is not sufficient. We hypothesise that this is related to the way, that the binning boundaries are chosen as well. The activation values in the layers closer to the input are smaller than the activation values in the layers closer to the output yielding a poor estimate of the distribution with the uniform binning strategy.
- The initialization method for Setting 1.1 and 1.2 is the same (it is not shifted into the positive domain for `ReLU` layers). Thus much of the distribution density will lie in the saturation region of the `ReLU` function, and since the bias is also initialized to zero, this could result in many of the neurons not firing in the network. If many of the neurons does not fire, it may be difficult to push the weights towards a desirable distribution in this case, as they will not get updated during backpropagation.

Based on the discussion above we decided to further investigate the effect of binning strategies and activation functions.

5.4 Further Investigation of Binning Strategies and Activation Functions

Based on the discussion in the Section 5.3.2 we will now perform a short analysis of whether or not, the violation of the DPI in Figure 5.3 is influenced by the maximum activation value observed. As seen in Figure 5.5 the maximum of the activation values can change significantly over the different runs.

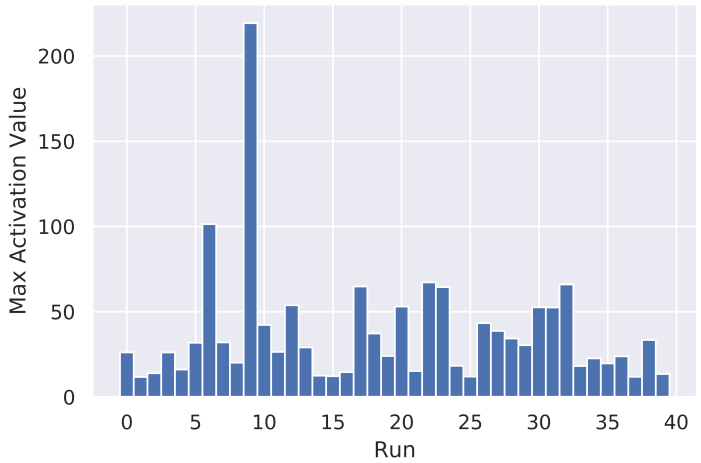
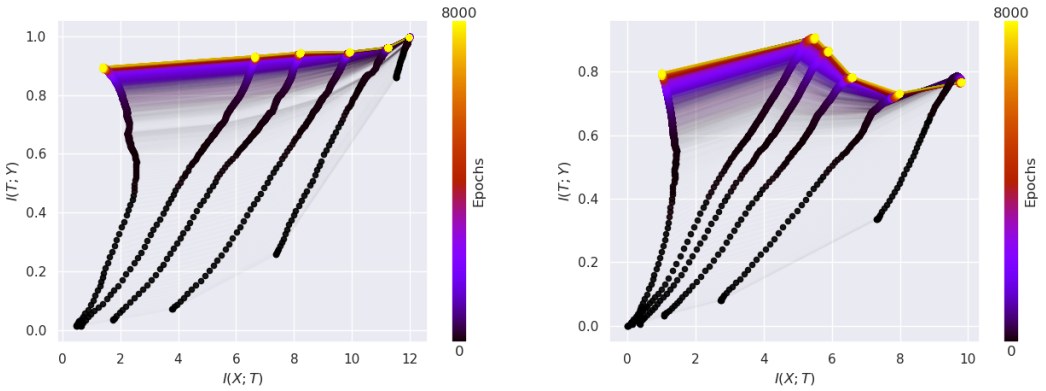


Figure 5.5: Visualization of the maximum activation values for Setting 1.2 with a mini-batch of size 256. The figure shows the maximum activation values for each of the 40 runs.

To see the effect of the maximum activation values and the information plane we sort the runs by the maximum activation value. We then plot the information plane averaged over the half of the trained networks with the smallest maximum activation value, and the half with the largest

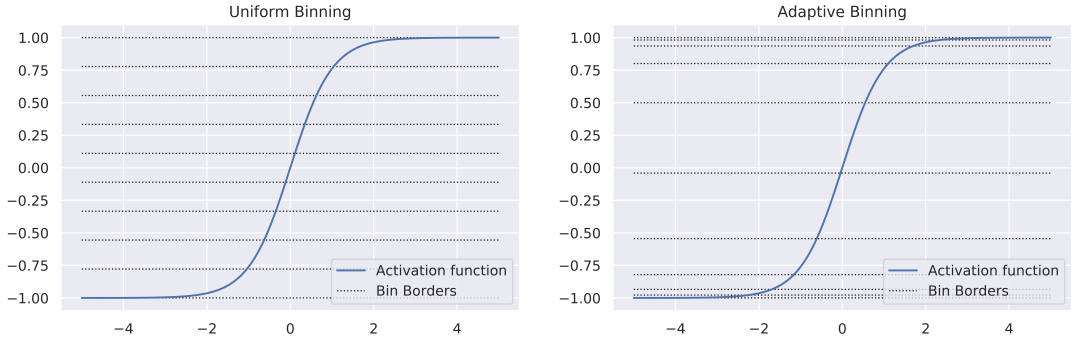
maximum activation values. As seen in Figure 5.6 the DPI breaks when the activation values get large, and the initial position of the layers cluster around the lower left corner.



(a) (b)

Figure 5.6: Visualization of the dynamics in the information plane as a consequence of the maximum value when trained with `ReLU`. (a) shows the information plane for the half of the runs with the smallest maximum activation values. (b) shows the result for the half of the runs with the largest activation values. It is clearly seen in (a) that the DPI breaks when the activation values are large.

A crucial aspect of these empirically observed properties thus seems to be *how* the mutual information is estimated. While Saxe et al. [16] show that the uniform binning strategy yields comparable results to other popular methods for estimating mutual information such as non-parametric kernel density estimators [12] and non-parametric k-NN based methods [13], recent work by Noshad et al. [14] and Chelombiev et al. [4] show that other methods based on either an *adaptive binning scheme* or dependence graphs yields different results. The authors suggest that the uniform binning methods underestimate the compression of the layers when unbounded activation functions are used.



(a) (b)

Figure 5.7: Visualization of uniform and adaptive binning strategy, for the same layer in the same network. Here 10 bins are used for illustrative purposes. Clearly much of the activation values concentrate around the saturation region as seen in (b).

We further investigate this claim by comparing the uniform binning scheme to an adaptive binning scheme (see Figure 5.7 for a visualization of how the bin width change): When using the uniform binning scheme for bounded activation functions the range of the binning is fixed

for all layers and all epochs. When using an unbounded activation function however the range is determined by taking the maximum value observed for all layers over all epochs. Thus many of the layers for possibly all but one of the epochs depend on a binning range that is inherently unrelated to the properties of most of the layers for most on the epochs. Rather it is related to the properties of that one layer with the largest activation value. Figure 5.8 shows the maximum activation value for each layer over 8000 epochs. Thus using this uniform binning scheme can result in many of the activation values being concentrated in just a few bins. [4]

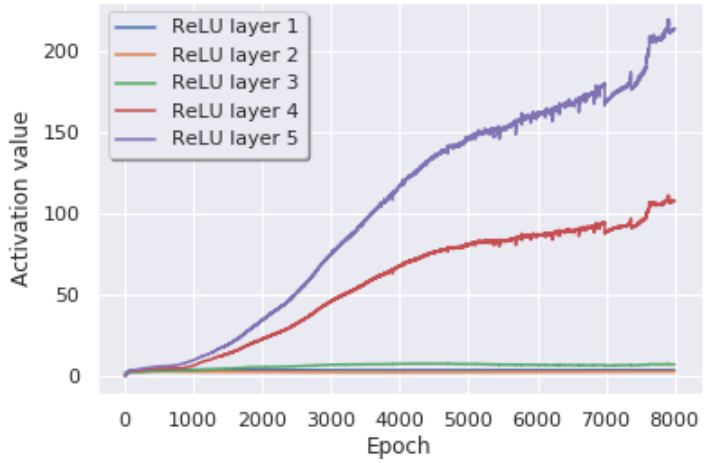


Figure 5.8: Visualization of the maximum activation values for Setting 1.2 with a mini-batch of size 256. The figure shows the maximum activation values for each of the ReLU layers after every epoch.

To avoid this phenomena we use an adaptive entropy based binning strategy [11] that will choose the bin boundaries for each layer in the network for each epoch. The bin boundaries are chosen such that each bin contains the same number of unique activation values. Given these bins, the computation of the mutual information for each hidden layer and the input and output is the same as for the uniform binning scheme. Using this strategy we avoid choosing a specific range, but rather let the range depend on the distribution of the activation values. Thus the method applies to all activation functions in the same way. [4]

We now investigate the effect of the binning strategy on the same network and activations functions as shown in the previous section, as well as a new unbounded activation function `ELU` and a new bounded activation function `ReLU6`. The following settings are now considered, where we use both the uniform and adaptive binning scheme this time with 30 bins *regardless* of the activation function. All settings use a batch size of 256.

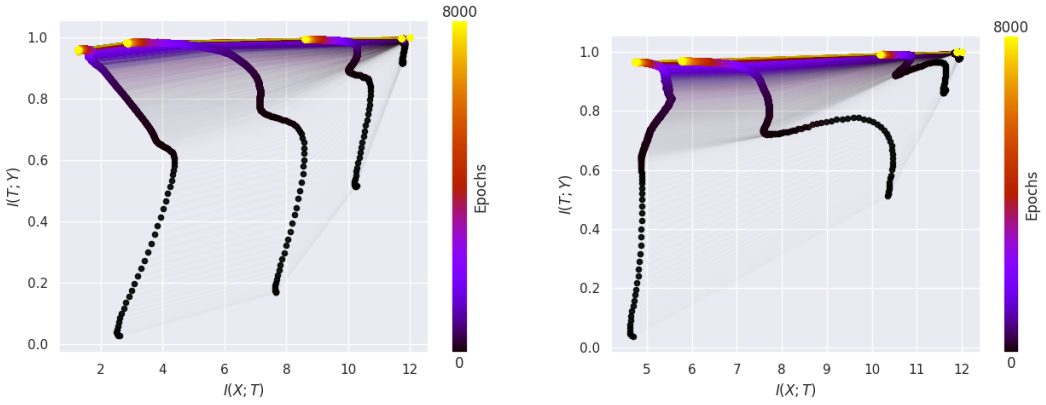
- **Setting 2.1:** Using $\tanh$ for all hidden layers except the last one where a `softmax` function is used. When using uniform binning, the range is $[-1, 1]$.
- **Setting 2.2:** Using `ReLU` for all hidden layers except the last one where a `softmax` function is used. When using uniform binning, the range is $[0, \max(\text{activation values})]$.
- **Setting 2.3:** Using `ELU` for all hidden layers except the last one where a `softmax` function is used. When using uniform binning, the range is $[-1, \max(\text{activation values})]$.

- **Setting 2.4:** Using `ReLU6` for all hidden layers except the last one where a `softmax` function is used. When using uniform binning, the range is $[0, 6]$.

5.4.1 Results

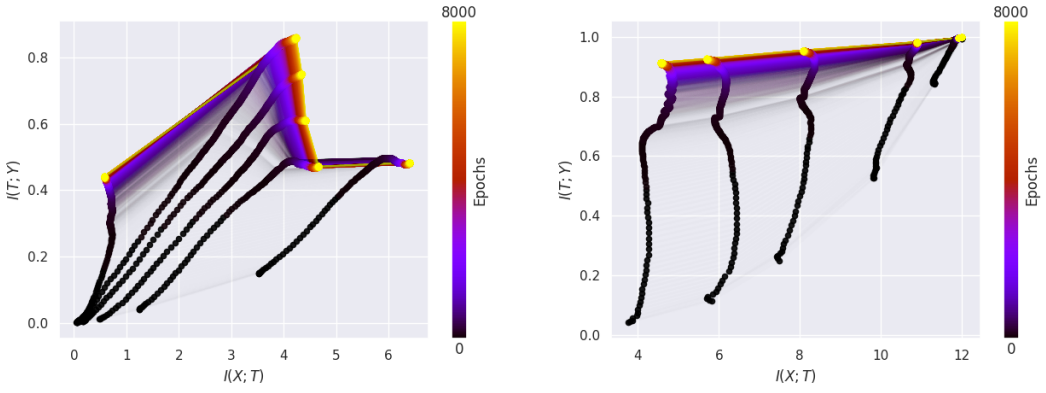
For **Setting 2.1** (see Figure 5.9) both methods show compression, however significant compression occurs in the initial fast phase indicated by a decrease in $I(X;T)$ for the adaptive binning scheme.

For **Setting 2.2** (see Figure 5.10) the information planes looks very different. The uniform binning strategy (a) with only 30 bins is insufficient for `ReLU` networks, as the DPI breaks here. Only a fitting phase seems to occur. The adaptive binning strategy (b) shows results where the DPI remains valid. Here the hidden layers end up in a similar position relative to $I(X;T)$ to where they started, cancelling each other out .



(a) (b)

Figure 5.9: **Setting 2.1**: Visualization of the information plane for the `IB network` using tanh for all hidden layers except the last where `softmax` function is used. (a) shows the uniform binning strategy, and (b) shows the adaptive binning strategy.

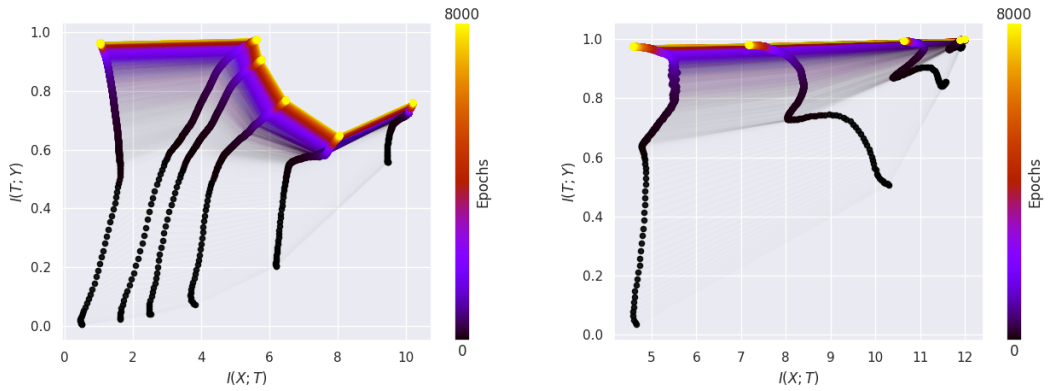


(a) (b)

Figure 5.10: **Setting 2.2**: Visualization of the information plane for the `IB network` using the `ReLU` activation function for all hidden layers except the last where `softmax` function is used. (a) shows the uniform binning strategy, and (b) shows the adaptive binning strategy.

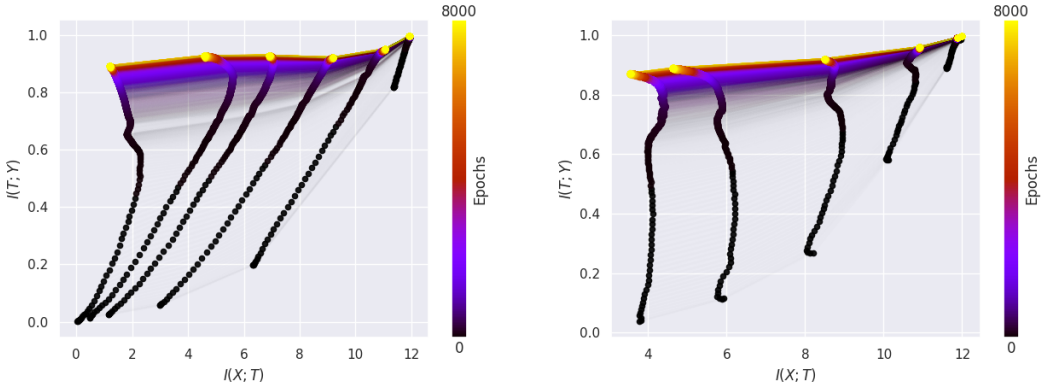
For **Setting 2.3** (see Figure 5.11), clearly the uniform binning strategy (b) with only 30 bins once again is insufficient as the DPI clearly breaks here. Interestingly the adaptive binning strategy (b) shows compression despite `ELU` being an unbounded activation function. It does however seem that the two phases are swapped, relative to the observations made by Shwartz-Ziv and Tishby [20]. During the initial fast phase we observe significant compression whereas the slower phase initially shows an increase in both $I(X;T)$ and $I(T;Y)$ indicating a fitting phase.

For **Setting 2.4** (see Figure 5.12) the information planes look very different, once again. This time the uniform binning strategy with 30 bins shows results comparable to Figure 5.3 where we used 100 bins and the `ReLU` activation function. This strengthens the hypothesis that the results from Figure 5.10 (a) and 5.11 (a) occur due to the magnitude of the activation values. It is interesting however that there is no observable compression phase even for the uniform binning strategy, since `ReLU6` is a saturating activation function. For the adaptive binning method the last two hidden layers show compression whereas for the other hidden layers the fitting and compression phase almost cancel each other out



(a) (b)

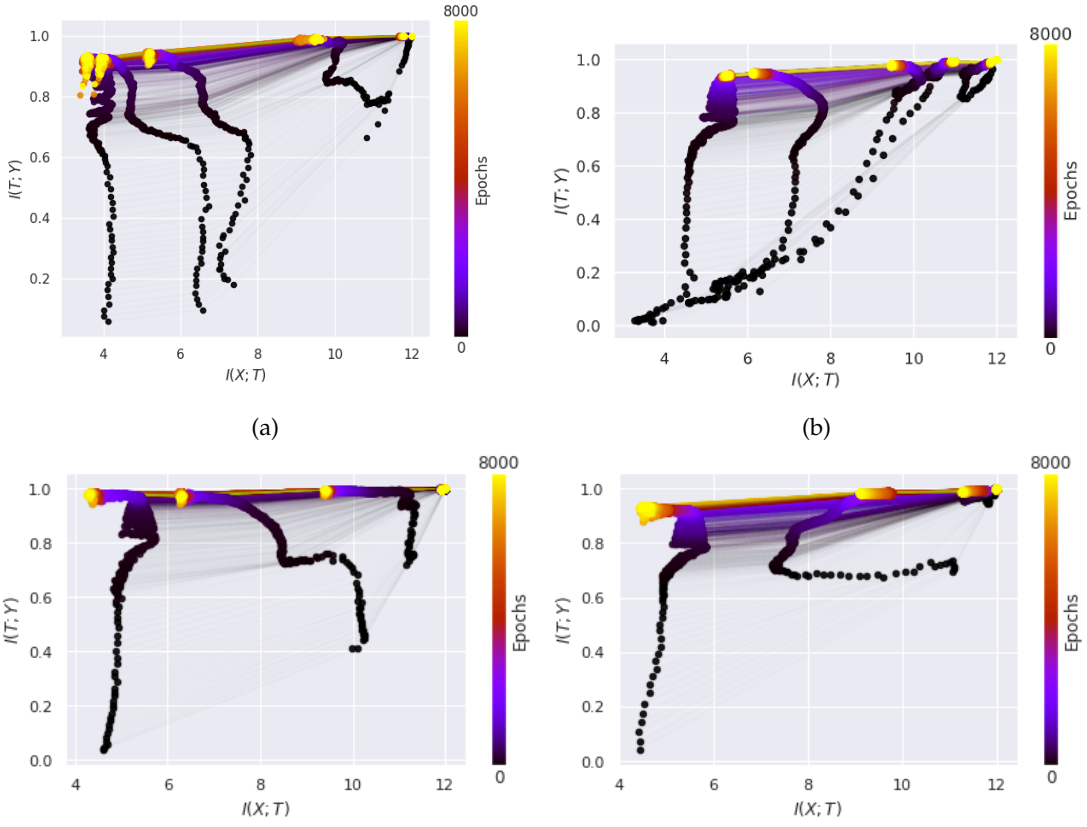
Figure 5.11: **Setting 2.3**: Visualization of the information plane for the `IB network` using the `ELU` activation function for all hidden layers except the last where `softmax` function is used.(a) shows the uniform binning strategy, and (b) shows the adaptive binning strategy



(a) (b)

Figure 5.12: **Setting 2.4**: Visualization of the information plane for the `IB network` using the `ReLU6` activation function for all hidden layers except the last where `softmax` function is used. (a) shows the uniform binning strategy, and (b) shows the adaptive binning strategy,

While running the experiments we observe that the information plane can differ significantly from one run to another, when using the adaptive binning strategy. Whether or not a network show compression, can seemingly be largely effected by the initialization of the weights.



(a) (b)

(c) (d)

Figure 5.13: (a, b) showing the difference in information plane dynamics between two runs of Setting 2.2. (c,d) shows the difference in the information plane between two runs of Setting 2.3. Only adaptive binning is used for these plots..

5.4.2 Discussion of results

Several of the observed results confirm the concerns raised in Section 5.3.2. We see that changing the number of bins to 30 still yields results that violate the DPI, whenever the activation function is unbounded and the uniform binning strategy is used. In fact we show, that using 30 bins rather than 100 show significantly different information planes, for the uniform binning strategy, despite Setting 2.2 and Setting 1.2 having the same network. For the adaptive binning strategy however the DPI remains valid for all activation functions we tested. Furthermore we see that, when using the adaptive binning strategy, the values do not, on average, cluster around the left corner of the information plane. This suggest that the uniform binning strategy underestimate the entropy, and consequently the mutual information of the hidden layers when using an unbounded activation function (see Appendix A for different network exhibiting the same phenomena). This is in correspondence with the results from [4]

Whether or not the adaptive binning strategy is accurate enough, does however need further analysis. We have yet to compare the estimated mutual information for an example where the true information can be computed. Thus the conclusion we can make only depends on whether the observations in the information plane satisfy the theory. It is also possible that, by changing the number of bins for the uniform binning scheme we can achieve comparable results to that of the adaptive binning scheme. We have yet to see any real concerns related to the estimation of mutual information for `tanh` layers, relative to the theory. It is possible that we could simply scale the number of bins according to the maximum value observed.

The question of how to choose the correct number of bins still remains for both methods as well. We have not paid much attention to it in this thesis, but rather chosen the numbers 30 and 100, as done in the literrature. While there exists suggestions on how to estimate this number [7, 23, 24] it seems like an unanswered question in the general case.

Lastly we highlight three specifically interesting points related to the claims outlined by Shwartz-Ziv and Tishby [20] and Saxe et al. [16], and the additional activation functions we tested in this section.

- We see that, while `ReLU6` is a bounded activation function no significant signs of a compression phase is visible for most of the hidden layers. It is possible however, that the lack of compression here is due to the activation values not concentrating around the saturation region.
- Even though `ELU` is an unbounded activation function the information plane looks very similar to the information plane for `tanh`. We note that both `tanh` and `ELU` have a lower bound in the negative domain (-1) and both of these settings show compression. Both `ReLU` and `ReLU6` show no significant signs of compression, when averaged over 40 runs, and are not defined in the negative domain. Whether or not this has any influence on the dynamics observed in the information plane requires further investigation. It is also possible that this is simply due to the problem of initialization strategy making some neurons not fire, as discussed in Section 5.3.2.
- In some cases e.g for Setting 2.3 shown in Figure 5.11 the compression phase and the fitting phase seems to have been swapped around. That is we first see a decrease in $I(T;X)$ followed by a slow phase where $I(T;X)$ and $I(T;Y)$ increase. This is inconsistent with the claims originally outlined by Schwartz-Ziv and Tishby [20] that the fitting phase is

causally linked to a high signal to noise ratio in the gradients, and that the compression phase is causally related to a low signal to noise ratio (noisy gradients).

Lastly Noshad et al. [14] claim specifically to observe compression in `ReLU` networks using their proposed method of estimating mutual information. This consequently debunk some of the claims by Saxe et al. [16]. However as we noted during our experiments; while `ReLU` networks can show compression it seems to largely depend on the initialization methods. The differences observed in our experiments relative to Noshad et al. [14] might simply be due to different initialization methods. Noshad et al. furthermore show the compression for a network of significantly different width than the one we considered, and on a different dataset. If we simply change the width of some of our hidden layers, we observe a compression and fitting phase for `ReLU` networks as well (see Appendix A and B).

6. Conclusion and Future Work

Since the beginning of the thesis we have come a long way. We started off accounting for the fundamentals of DNNs and relevant quantities from information theory. We then gave an overview of rate distortion theory and the information bottleneck method and discussed its connection to DNNs. Specifically how the hidden layers in a feedforward neural network can be quantified by a point $(I(X;T), I(T;Y))$ in the information plane.

After this we reproduced some key experiments from the original papers by Shwartz-Ziv and Tishby [20] and Saxe et al. [16]. We found that; (i) Using a uniform binning strategy compression was only observed in the setting with the double saturating `tanh` activation function. (ii) When changing the activation function to the unbounded `ReLU` function only the error-minimization phase was observed. (iii) While this was in correspondence with the claims outlined in the paper by Saxe et al. [16] and cast doubts to the claims by Shwartz-Ziv and Tishby [20], we observed several potential issues linked to using a uniform binning scheme for unbounded activation functions.

These issues were further investigated and confirmed by observing that; (i) The DPI breaks when the magnitude of the activation values are large. (ii) The initial position of the hidden layers cluster around the left corner of the information plane. We decided to rerun the experiments with a new adaptive binning strategy and additional activation functions. We found that the aforementioned issues with the uniform binning strategy was not present using this new method. We furthermore observed that, in correspondence with Noshad et al. [14] and Chelombiev et al. [4], the uniform binning strategy seems to underestimate the mutual information for unbounded activation functions. Using the adaptive binning strategy we observed the following; (i) Different methods of mutual information estimation can yield drastically different information plane dynamics for the same network. (ii) Compression is not limited to networks with double saturating activation functions, and can occur in networks with unbounded activation functions as well. (iii) Similar activation functions can produce vastly different paths for the hidden layers in the information plane. (vi) Activation functions defined, not only in the positive domain, seem to compress information significantly more, on average, compared to activation functions defined over only the positive real domain. However this might simply be due to initialization of the weights, as we saw that initialization of weights have a drastic influence on the trajectory of the hidden layers.

In light of these conclusions we propose a number of interesting research topics for the future of IB theory and deep learning.

- While the adaptive binning method obey the limitations of the information plane imposed by the theory, the method is still not rigorously derived. The number of bins seems to be less important, as the bin widths are adaptive, however the problem of choosing a good

number of bins still remains, although not explored in this thesis. It would be interesting to see experiments on setups where the mutual information can be computed exactly, comparing different methods of estimating mutual information in high dimensional spaces to the true mutual information.

- The effect of parameter choices and initialization methods remains unexplored, albeit a large factor relative to the compression, as seen in our experiments. Under the assumption, that the compression is causally related to generalization of a network, this could potentially lead to new initialization strategies. The optimality of the choice of parameters such as the learning rate, batch size and activation function can be evaluated relative to the information plane.

- Unrelated to the information plane it has been established that two signal-to-noise phases occurs during the training phase (see eg. [3]), specifically as the loss drops quickly during the first epochs the signal-to-noise ratio is high, that is the mean of the gradients is large relative to the standard deviation of the gradients. After the loss saturates a low signal-to-noise phase occurs. Shwartz-Ziv and Tishby [20] claim that the fitting and compression phase is casually linked to these two signal-to-noise phases of the training process. As we concluded, the two phases can be swapped (fitting and compression). This seem to go directly against the claim that the compression is linked to the noisy low signal-to-noise ratio. Experiments directly confirming this by plotting the signal-to-noise would be interesting, and only require few changes to the codebase of this thesis.

- While these experiments produce interesting results, and opens up for a needed discussion on the theory of deep learning, there is still a lack of experiments on larger real world datasets. The computational bottleneck when performing these experiments is the computation of the mutual information. Saving the activation values for all the hidden layers takes up large amounts of memory, and the computation of the mutual information is computationally more expensive than training the networks. Algorithms for computing mutual information in an computationally efficient manner compared to current strategies are thus sought after.

Bibliography

[1] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[2] Ovidiu Calin. *Deep learning architectures: a mathematical approach*. Springer, 2020.

[3] Jerry Chee and Panos Toulis. Convergence diagnostics for stochastic gradient descent with constant learning rate. In *International Conference on Artificial Intelligence and Statistics*, pages 1476–1485, 2018.

[4] Ivan Chelombiev, Conor Houghton, and Cian O'Donnell. Adaptive estimators show information compression in deep neural networks. *arXiv preprint arXiv:1902.09037*, 2019.

[5] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006. ISBN 0471241954.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] David Freedman and Persi Diaconis. On the histogram as a density estimator: L 2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4):453–476, 1981.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[9] Stanislaw Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Width of minima reached by stochastic gradient descent is influenced by learning rate to batch size ratio. In *International Conference on Artificial Neural Networks*, pages 392–402. Springer, 2018.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Ron Kohavi and Mehran Sahami. Error-based and entropy-based discretization of continuous features. In *KDD*, pages 114–119, 1996.

[12] Artemy Kolchinsky and Brendan D Tracey. Estimating mixture entropy with pairwise distances. *Entropy*, 19(7):361, 2017.

[13] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

[14] Morteza Noshad, Yu Zeng, and Alfred O Hero. Scalable mutual information estimation using dependence graphs. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2962–2966. IEEE, 2019.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[16] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

[17] Ohad Shamir, Sivan Sabato, and Naftali Tishby. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29-30):2696–2711, 2010.

[18] Claude E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[19] Claude E Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE Nat. Conv. Rec*, 4(142-163):1, 1959.

[20] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[21] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[22] Noam Slonim. *The information bottleneck: Theory and applications*. PhD thesis, Hebrew University of Jerusalem, 2002.

[23] Herbert A Sturges. The choice of a class interval. *Journal of the american statistical association*, 21(153):65–66, 1926.

[24] George R Terrell and David W Scott. Oversmoothed nonparametric density estimates. *Journal of the American Statistical Association*, 80(389):209–214, 1985.

[25] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2015.

[26] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

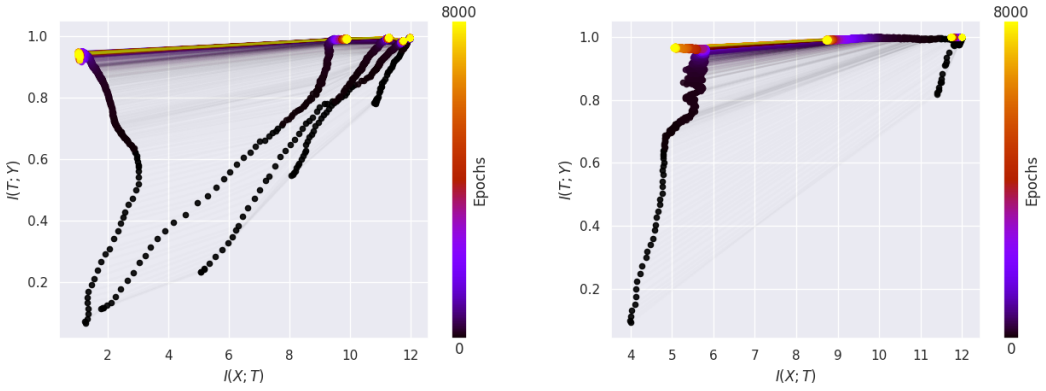
[27] Natalie Wolchover and Lucy Reading. New theory cracks open the black box of deep learning. *Quanta Magazine*, 3, 2017.

[28] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

A. Linear Network

Below we consider the information plane for a network with layer sizes of $12-12-12-12-12-2$ using `ReLU` with both a uniform binning strategy and an adaptive strategy. We only average over 2 runs of each setting. We consider 2 settings.

- **Setting A.1 :** Here the `ReLU` activation is used for all hidden layers except for the last where a softmax is used. Here we use a uniform binning scheme with 100 bins.
- **Setting A.2 :** Here the `ReLU` activation is used for all hidden layers except for the last where a softmax is used. Here we use a adaptive binning scheme with 30 bins.



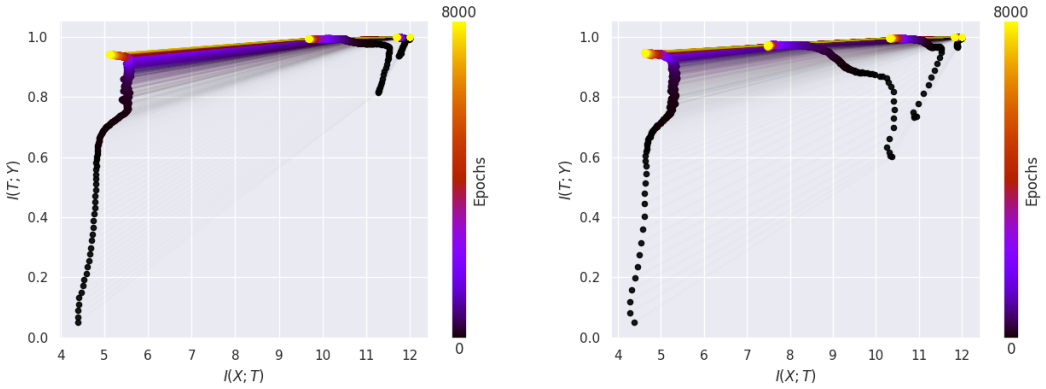
(a) (b)

Figure A.1: Visualization of information planes for a different network exhibiting vastly different trajectory of the hidden layers depended on the binning strategy (a) shows the information plane for Setting A.1 with a uniform binning scheme with 100 bins for the `ReLU` activation function. (b) shows the information plane for Setting A.2 with the `ReLU` activation function and the adaptive binning strategy.

B. ReLU Compression Networks

Below we consider the information plane for a networks different from the `IB network` and som that `ReLU` layers can compress on average. Both settings using `ReLU` for all hidden layers except the last layer where a `softmax` function is used. We use the adaptive binning strategy with 30 bins. The information plane is averaged over 30 runs of the same network with random weight initialization and train-test split.

- **Setting B.1 :** Network with layer sizes $12 - 12 - 12 - 10 - 8 - 2$ is used.
- **Setting B.2 :** Network with layer sizes $12 - 12 - 12 - 10 - 8 - 7 - 2$ is used.



(a) (b)

Figure B.1: Visualization of the information plane for networks where `ReLU` shows compression (a) shows the information plane for Setting B.1 with the adaptive binning scheme. All layers `ReLU` layers except the last where `softmax` is used. (b) shows the information plane for Setting B.2 with the adaptive binning scheme. All layers `ReLU` layers except the last where `softmax` is used.

C. Code

The written for the thesis can be found here: `https://github.com/lrnq/ibbsc`